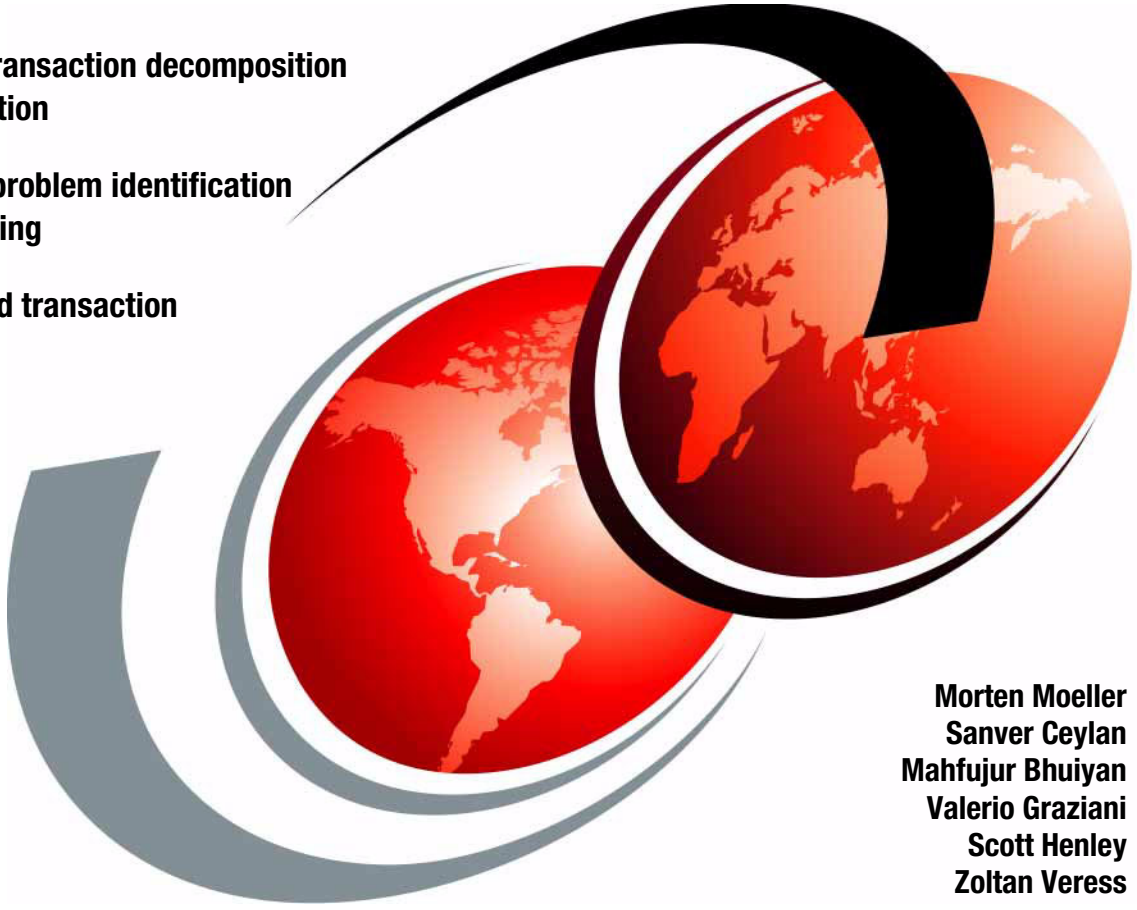
Tivoli software

IBM

End-to-End e-business Transaction Management Made Easy

- Seamless transaction decomposition and correlation
- Automatic problem identification and baselining
- Policy based transaction discovery



Morten Moeller
Sanver Ceylan
Mahfujur Bhuiyan
Valerio Graziani
Scott Henley
Zoltan Veress

Redbooks

ibm.com/redbooks

International Technical Support Organization

End-to-End e-business Transaction Management Made Easy

December 2003

SG24-6080-00

Note: Before using this information and the product it supports, read the information in “Notices” on page xix.

First Edition (December 2003)

This edition applies to Version 5, Release 2 of IBM Tivoli Monitoring for Transaction Performance (product number 5724-C02).

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this redbook for more current information.

© Copyright International Business Machines Corporation 2003. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures ix

Tables xvii

Notices xix
Trademarks xx

Preface xxi
The team that wrote this redbook. xxii
Become a published author xxiv
Comments welcome. xxiv

Part 1. Business value of end-to-end transaction monitoring 1

Chapter 1. Transaction management imperatives 3
1.1 e-business transactions. 4
1.2 J2EE applications management 5
1.2.1 The impact of J2EE on infrastructure management 7
1.2.2 Importance of JMX 8
1.3 e-business applications: complex layers of services. 11
1.3.1 Managing the e-business applications 15
1.3.2 Architecting e-business application infrastructures. 21
1.3.3 Basic products used to facilitate e-business applications 23
1.3.4 Managing e-business applications using Tivoli. 26
1.4 Tivoli product structure 28
1.5 Managing e-business applications 32
1.5.1 IBM Tivoli Monitoring for Transaction Performance functions. 33

Chapter 2. IBM Tivoli Monitoring for Transaction Performance in brief. . 37
2.1 Typical e-business transactions are complex 38
2.1.1 The pain of e-business transactions 38
2.2 Introducing TMTP 5.2 40
2.2.1 TMTP 5.2 components 40
2.3 Reporting and troubleshooting with TMTP WTP. 44
2.4 Integration points. 51

Chapter 3. IBM TMTP architecture. 55
3.1 Architecture overview 56
3.1.1 Web Transaction Performance 56
3.1.2 Enterprise Transaction Performance 58

© Copyright IBM Corp. 2003. All rights reserved.

3.2 Physical infrastructure components 61
3.3 Key technologies utilized by WTP 67
3.3.1 ARM 67
3.3.2 J2EE instrumentation 72
3.4 Security features 76
3.5 TMTP implementation considerations 79
3.6 Putting it all together 80

Part 2. Installation and deployment 83

Chapter 4. TMTP WTP Version 5.2 installation and deployment 85
4.1 Custom installation of the Management Server 87
4.1.1 Management Server custom installation preparation steps 88
4.1.2 Step-by-step custom installation of the Management Server 107
4.1.3 Deployment of the Store and Forward Agents 118
4.1.4 Installation of the Management Agents 130
4.2 Typical installation of the Management Server 137

Chapter 5. Interfaces to other management tools 153
5.1 Managing and monitoring your Web infrastructure 154
5.1.1 Keeping Web and application servers online 154
5.1.2 ITM for Web Infrastructure installation 155
5.1.3 Creating managed application objects 158
5.1.4 WebSphere monitoring 162
5.1.5 Event handling 168
5.1.6 Surveillance: Web Health Console 170
5.2 Configuration of TEC to work with TMTP 171
5.2.1 Configuration of ITM Health Console to work with TMTP 173
5.2.2 Setting SNMP 175
5.2.3 Setting SMTP 176

Chapter 6. Keeping the transaction monitoring environment fit 177
6.1 Basic maintenance for the TMTP WTP environment 178
6.1.1 Checking MBeans 182
6.2 Configuring the ARM Agent 184
6.3 J2EE monitoring maintenance 188
6.4 TMTP TDW maintenance tips 191
6.5 Uninstalling the TMTP Management Server 193
6.5.1 The right way to uninstall on UNIX 193
6.5.2 The wrong way to uninstall on UNIX 195
6.5.3 Removing GenWin from a Management Agent 195
6.5.4 Removing the J2EE component manually 196
6.6 TMTP Version 5.2 best practices 204

Part 3. Using TMTP to measure transaction performance 209

Chapter 7. Real-time reporting 211
7.1 Reporting overview 212
7.2 Reporting differences from Version 5.1 212
7.3 The Big Board 213
7.4 Topology Report overview 215
7.5 STI Report 219
7.6 General Reports 219

Chapter 8. Measuring e-business transaction response times 225
8.1 Preparation for measurement and configuration 227
8.1.1 Naming standards for TMTP policies 228
8.1.2 Choosing the right measurement component(s) 229
8.1.3 Measurement component selection summary 234
8.2 The sample e-business application: Trade 235
8.3 Deployment, configuration, and ARM data collection 239
8.4 STI recording and playback 241
8.4.1 STI component deployment 241
8.4.2 STI Recorder installation 242
8.4.3 Transaction recording and registration 245
8.4.4 Playback schedule definition 248
8.4.5 Playback policy creation 251
8.4.6 Working with realms 255
8.5 Quality of Service 257
8.5.1 QoS Component deployment 259
8.5.2 Creating discovery policies for QoS 261
8.6 The J2EE component 278
8.6.1 J2EE component deployment 278
8.6.2 J2EE component configuration 282
8.7 Transaction performance reporting 295
8.7.1 Reporting on Trade 296
8.7.2 Looking at subtransactions 297
8.7.3 Using topology reports 300
8.8 Using TMTP with BEA Weblogic 307
8.8.1 The Java Pet Store sample application 308
8.8.2 Deploying TMTP components in a Weblogic environment 310
8.8.3 J2EE discovery and listening policies for Weblogic Pet Store 312
8.8.4 Event analysis and online reports for Pet Store 316

Chapter 9. Rational Robot and GenWin 325
9.1 Introducing Rational Robot 326
9.1.1 Installing and configuring the Rational Robot 326
9.1.2 Configuring a Rational Project 339

9.1.3 Recording types: GUI and VU scripts 344
9.1.4 Steps to record a GUI simulation with Rational Robot 345
9.1.5 Add ARM API calls for TMTP in the script 351
9.2 Introducing GenWin 365
9.2.1 Deploying the Generic Windows Component 365
9.2.2 Registering your Rational Robot Transaction 368
9.2.3 Create a GenWin playback policy 369

Chapter 10. Historical reporting 375
10.1 TMTP and Tivoli Enterprise Data Warehouse 376
10.1.1 Tivoli Enterprise Data Warehouse overview 376
10.1.2 TMTP Version 5.2 Warehouse Enablement Pack overview 380
10.1.3 The monitoring process data flow 382
10.1.4 Setting up the TMTP Warehouse Enablement Packs 383
10.2 Creating historical reports directly from TMTP 405
10.3 Reports by TEDW Report Interface 406
10.3.1 The TEDW Report Interface 406
10.3.2 Sample TMTP Version 5.2 reports with data mart 408
10.3.3 Create extreme case weekly and monthly reports 413
10.4 Using OLAP tools for customized reports 417
10.4.1 Crystal Reports overview 418
10.4.2 Crystal Reports integration with TEDW 418
10.4.3 Sample Trade application reports 421

Part 4. Appendixes 427

Appendix A. Patterns for e-business 429
Introduction to Patterns for e-business 430
The Patterns for e-business layered asset model 431
How to use the Patterns for e-business 433

Appendix B. Using Rational Robot in the Tivoli Management Agent environment 439
Rational Robot 440
Tivoli Monitoring for Transaction Performance (TMTP) 440
The ARM API 441
Initial install 443
Working with Java Applets 449
Running the Java Enabler 450
Using the ARM API in Robot scripts 450
Rational Robot command line options 462
Obfuscating embedded passwords in Rational Scripts 464
Rational Robot screen locking solution 468

Appendix C. Additional material 473
Locating the Web material 473
Using the Web material 473
System requirements for downloading the Web material 474
How to use the Web material 474

Abbreviations and acronyms 475

Related publications 479
IBM Redbooks 479
Other resources 480
Referenced Web sites 481
How to get IBM Redbooks 482
IBM Redbooks collections 482
Help from IBM 482

Index 483

Figures

1-1 Transaction breakdown 4
1-2 Growing infrastructure complexity 12
1-3 Layers of service 14
1-4 The ITIL Service Management disciplines 17
1-5 Key relationships between Service Management disciplines 20
1-6 A typical e-business application infrastructure 21
1-7 e-business solution-specific service layers 24
1-8 Logical view of an e-business solution 25
1-9 Typical Tivoli-managed e-business application infrastructure 27
1-10 The On Demand Operating Environment 28
1-11 IBM Automation Blueprint 30
1-12 Tivoli's availability product structure 31
1-13 e-business transactions 34
2-1 Typical e-business transactions are complex 38
2-2 Application topology discovered by TMTP 42
2-3 Big Board View 44
2-4 Topology view indicating problem 45
2-5 Inspector view 46
2-6 Instance drop down 46
2-7 Instance topology 47
2-8 Inspector viewing metrics 48
2-9 Overall Transactions Over Time 49
2-10 Transactions with Subtransactions 50
2-11 Page Analyzer Viewer 50
2-12 Launching the Web Health Console from the Topology view 51
3-1 TMTP Version 5.2 architecture 56
3-2 Enterprise Transaction Performance architecture 60
3-3 Management Server architecture 62
3-4 Requests from Management Agent to Management Server via SOAP . 63
3-5 Management Agent JMX architecture 64
3-6 ARM Engine communication with Monitoring Engine 66
3-7 Transaction performance visualization 69
3-8 Tivoli Just-in-Time Instrumentation overview 75
3-9 SnF Agent communication flows 78
3-10 Putting it all together 81
4-1 Customer production environment 87
4-2 WebSphere information screen 92
4-3 ikeyman utility 93

© Copyright IBM Corp. 2003. All rights reserved.

4-4 Creation of custom JKS file 94
4-5 Set password for the JKS file 94
4-6 Creating a new self signed certificate 95
4-7 New self signed certificate options 96
4-8 Password change of the new self signed certificate 97
4-9 Modifying self signed certificate passwords 97
4-10 GSKit new KDB file creation 99
4-11 CMS key database file creation 99
4-12 Password setup for the prodsnf.kdb 100
4-13 New Self Signed Certificate menu 100
4-14 Create new self signed certificate 101
4-15 Trust files and certificates 102
4-16 The imported certificates 103
4-17 Extract Certificate 104
4-18 Extracting certificate from the msprod.jks file 104
4-19 Add a new self signed certificate 105
4-20 Adding a new self signed certificate 105
4-21 Label for the certificate 106
4-22 The imported self signed certificate 106
4-23 Welcome screen on the Management Server installation wizard 108
4-24 License agreement panel 109
4-25 Installation target folder selection 110
4-26 SSL enablement window 111
4-27 WebSphere configuration panel 112
4-28 Database options panel 113
4-29 Database Configuration panel 114
4-30 Setting summarization window 115
4-31 Installation progress window 116
4-32 The finished Management Server installation 117
4-33 TMTP logon window 118
4-34 Welcome window of the Store and Forward agent installation 119
4-35 License agreement window 120
4-36 Installation location specification 121
4-37 Configuration of Proxy host and mask window 122
4-38 KDB file definition 123
4-39 Communication specification 124
4-40 User Account specification window 125
4-41 Summary before installation 126
4-42 Installation progress 127
4-43 The WebSphere caching proxy reboot window 128
4-44 The final window of the installation 129
4-45 Management Agent installation welcome window 130
4-46 License agreement window 131

4-47 Installation location definition 132
4-48 Management Agent connection window 133
4-49 Local user account specification 134
4-50 Installation summary window 135
4-51 The finished installation 136
4-52 Management Server Welcome screen 138
4-53 Management Server License Agreement panel 139
4-54 Installation location window 140
4-55 SSL enablement window 141
4-56 WebSphere Configuration window 142
4-57 Database options window 143
4-58 DB2 administrative user account specification 144
4-59 User specification for fenced operations in DB2 145
4-60 User specification for the DB2 instance 146
4-61 Management Server installation progress window 147
4-62 DB2 silent installation window 148
4-63 WebSphere Application Server silent installation 149
4-64 Configuration of the Management Server 150
4-65 The finished Management Server installation 151
5-1 Create WSAdministrationServer 159
5-2 Create WSApplicationServer 160
5-3 Discover WebSphere Resources 161
5-4 WebSphere managed application object icons 162
5-5 Example for an IBM Tivoli Monitoring Profile 167
5-6 Web Health Console using WebSphere Application Server 171
5-7 Configure User Setting for ITM Web Health Console 174
6-1 WebSphere started without sourcing the DB2 environment 179
6-2 Management Server ping output 180
6-3 MBean Server HTTP Adapter 183
6-4 Duplicate row at the TWH_CDW 192
6-5 Rational Project exists error message 196
6-6 WebSphere 4 Admin Console 197
6-7 Removing the JVM Generic Arguments 199
6-8 WebLogic class path and argument settings 202
6-9 Configuring the J2EE Trace Level 206
6-10 Configuring the Sample Rate and Failure Instances collected 207
7-1 The Big Board 214
7-2 Topology Report 216
7-3 Node context reports 217
7-4 Topology Line Chart 218
7-5 STI Reports 219
7-6 General reports 220
7-7 Transactions with Subtransactions report 221

7-8 Availability graph 222
7-9 Page Analyzer Viewer 223
8-1 Trade3 architecture 236
8-2 WAS 5.0 Admin console: Install of Trade3 application 238
8-3 Deployment of STI components. 242
8-4 STI Recorder setup welcome dialog 243
8-5 STI Software License Agreement dialog 243
8-6 Installation of STI Recorder with SSL disable 244
8-7 installation of STI Recorder with SSL enabled. 244
8-8 STI Recorder is recording the Trade application 246
8-9 Creating STI transaction for trade 247
8-10 Application steps run by trade_2_stock-check playback policy 248
8-11 Creating a new playback schedule 249
8-12 Specify new playback schedule properties 250
8-13 Create new Playback Policy 251
8-14 Configure STI Playback 252
8-15 Assign name to STI Playback Policy 255
8-16 Specifying realm settings 256
8-17 Proxies in an Internet environment 258
8-18 Work with agents QoS 259
8-19 Deploy QoS components 260
8-20 Work with Agents: QoS installed 261
8-21 Multiple QoS systems measuring multiple sites. 265
8-22 Work with discovery policies 267
8-23 Configure QoS discovery policy. 268
8-24 Choose schedule for QoS 269
8-25 Selecting Agent Group for QoS discovery policy deployment 270
8-26 Assign name to new QoS discovery policy 271
8-27 View discovered transactions to define QoS listening policy 272
8-28 View discovered transaction of trade application. 273
8-29 Configure QoS set data filter: write data 274
8-30 Configure QoS automatic threshold. 275
8-31 Configure QoS automatic threshold for Back-End Service Time 276
8-32 Configure QoS and assign name. 277
8-33 Deploy J2EE and Work of agents 279
8-34 J2EE deployment and configuration for WAS 5.0.1. 280
8-35 J2EE deployment and work with agents 282
8-36 J2EE: Work with Discovery Policies 283
8-37 Configure J2EE discovery policy 284
8-38 Work with Schedules for discovery policies 285
8-39 Assign Agent Groups to J2EE discovery policy 286
8-40 Assign name J2EE. 287
8-41 Create a listening policy for J2EE 289

8-42 Creating listening policies and selecting application transactions 290
8-43 Configure J2EE listener 291
8-44 Configure J2EE parameter and threshold for performance 292
8-45 Assign a name for the J2EE listener 295
8-46 Event Graph: Topology view for Trade application 297
8-47 Trade transaction and subtransaction response time by STI. 298
8-48 Back-End service Time for Trade subtransaction 3 299
8-49 Time used by servlet to perform Trade back-end process. 300
8-50 STI topology relationship with QoS and J2EE 301
8-51 QoS Inspector View from topology correlation with STI and J2EE . . . 302
8-52 Response time view of QoS Back end service(1) time 303
8-53 Response time view of Trade application relative to threshold 304
8-54 Trade EJB response time view get market summary() 305
8-55 Topology view of J2EE and trade JDBC components 306
8-56 Topology view of J2EE details Trade EJB: get market summary() . . . 307
8-57 Pet Store application welcome page 309
8-58 Weblogic 7.0.1 Admin Console 310
8-59 Weblogic Management Agent configuration 311
8-60 Creating listening policy for Pet Store J2EE Application 313
8-61 Choose Pet Store transaction for Listening policy 314
8-62 Automatic threshold setting for Pet Store 314
8-63 QoS listening policies for Pet Store automatic threshold setting 315
8-64 QoS correlation with J2EE application. 316
8-65 Pet Store transaction and subtransaction response time by STI 317
8-66 Page Analyzer Viewer report of Pet Store business transaction 318
8-67 Correlation of STI and J2EE view for Pet Store application. 319
8-68 J2EE dofilter() methods creates events 320
8-69 Problem indication in topology view of Pet Store J2EE application . . . 321
8-70 Topology view: event violation by getShoppingClientFacade 322
8-71 Response time for getShoppingClienFacade method 322
8-72 Real-time Round Trip Time and Back-End Service Time by QoS 323
9-1 Rational Robot Install Directory 327
9-2 Rational Robot installation progress 328
9-3 Rational Robot Setup wizard 328
9-4 Select Rational Robot component 329
9-5 Rational Robot deployment method. 329
9-6 Rational Robot Setup Wizard. 330
9-7 Rational Robot product warnings. 330
9-8 Rational Robot License Agreement 331
9-9 Destination folder for Rational Robot 331
9-10 Ready to install Rational Robot 332
9-11 Rational Robot setup complete 332
9-12 Rational Robot license key administrator wizard 333

9-13 Import Rational Robot license 334
9-14 Import Rational Robot license (cont...) 334
9-15 Rational Robot license imported successfully 334
9-16 Rational Robot license key now usable 335
9-17 Configuring the Rational Robot Java Enabler 336
9-18 Select appropriate JVM 337
9-19 Select extensions 338
9-20 Rational Robot Project 340
9-21 Configuring project password 341
9-22 Finalize project 342
9-23 Configuring Rational Project 343
9-24 Specifying project datastore 344
9-25 Record GUI Dialog Box 346
9-26 GUI Insert 346
9-27 Verification Point Name Dialog 348
9-28 Object Finder Dialog 349
9-29 Object Properties Verification Point panel 350
9-30 Debug menu 354
9-31 GUI Playback Options 355
9-32 Entering the password for use in Rational Scripts 358
9-33 Terminal Server Add-On Component 361
9-34 Setup for Terminal Server client 362
9-35 Terminal Client connection dialog 363
9-36 Start Browser Dialog 364
9-37 Deploy Generic Windows Component 366
9-38 Deploy Components and/or Monitoring Component 367
9-39 Work with Transaction Recordings 368
9-40 Create Generic Windows Transaction 369
9-41 Work with Playback Policies 370
9-42 Configure Generic Windows Playback 370
9-43 Configure Generic Windows Thresholds 371
9-44 Choosing a schedule 372
9-45 Specify Agent Group 373
9-46 Assign your playback policy a name 374
10-1 A typical TEDW environment 378
10-2 TMTP Version 5.2 warehouse data model 381
10-3 ITMTP: Enterprise Transaction Performance data flow 382
10-4 Tivoli Enterprise Data Warehouse installation scenario 383
10-5 TEDW installation 388
10-6 TEDW installation type 388
10-7 TEDW installation: DB2 configuration 389
10-8 Path to the installation media for the ITM Generic ETL1 program 389
10-9 TEDW installation: Additional modules 390

10-10 TMTP ETL1 and ETL2 program installation 390
10-11 TEDW installation: Installation running 391
10-12 Installation summary window 391
10-13 TMTP ETL Source and Target 395
10-14 BWB_TMTP_DATA_SOURCE user ID information 396
10-15 Warehouse source table properties 397
10-16 TableSchema and TableName for TMTP Warehouse sources 398
10-17 Warehouse source table names changed 398
10-18 Warehouse source table names immediately after installation 399
10-19 Scheduling source ETL process 402
10-20 Scheduling soure ETL process periodically 403
10-21 Source ETL scheduled processes to Production status 405
10-22 Pet Store STI transaction response time report for eight days 406
10-23 Response time by Application 409
10-24 Response time by host name 410
10-25 Execution Load by Application daily 411
10-26 Performance Execution load by User 412
10-27 Performance Transaction availability% Daily 413
10-28 Add metrics window 415
10-29 Add Filter windows 416
10-30 Weekly performance load execution by user for trade application . . . 417
10-31 Create links for report generation in Crystal Reports 419
10-32 Choose fields for report generation 420
10-33 Crystal Reports filtering definition 421
10-34 trade_2_stock-check_tivlab01 playback policy end-user experience . 422
10-35 trade_j2ee_lis listening policy response time report 423
10-36 Response time JDBC process: Trade applications executeQuery() . . 424
10-37 Response time for trade by trade_qos_lis listening policy 425
A-1 Patterns layered asset model 432
A-2 Pattern representation of a Custom design 434
A-3 Custom design 435
B-1 ETP Average Response Time 441
B-2 ARM API Calls 442
B-3 Rational Robot Project Directory 443
B-4 Rational Robot Project 444
B-5 Rational Robot Project 445
B-6 Configuring project password 446
B-7 Finalize project 447
B-8 Configuring Rational Project 448
B-9 Specifying project datastore 449
B-10 Scheduler 454
B-11 Scheduling wizard 455
B-12 Scheduler frequency 456

B-13 Schedule start time 457
B-14 Schedule user 458
B-15 Select schedule advanced properties 459
B-16 Enable scheduled task 460
B-17 Viewing schedule frequency 461
B-18 Advanced scheduling options 462
B-19 Entering the password for use in Rational Scripts 466
B-20 Terminal Server Add-On Component 469
B-21 Setup for Terminal Server client 470
B-22 Terminal Client Connection Dialog 471

Tables

4-1 File system creation 89
4-2 JKS file creation differences 98
4-3 Internet Zone SnF different parameters 129
4-4 Changed option of the Management Agent installation/zone 136
5-1 Minimum monitoring levels WebSphere Application Server 157
5-2 Resource Model indicator defaults 164
6-1 ARM engine log levels 185
7-1 Big Board Icons 214
8-1 Choosing monitoring components 234
8-2 J2EE components configuration properties 281
8-3 Pet Store J2EE configuration parameters 311
10-1 Measurement codes 387
10-2 Source database names used by the TMTP ETLs 393
10-3 Warehouse processes 401
10-4 Warehouse processes and components 404
A-1 Business patterns 433
A-2 Integration patterns 434
A-3 Composite patterns 435
B-1 Rational Robot command line options 462

© Copyright IBM Corp. 2003. All rights reserved.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

© Copyright IBM Corp. 2003. All rights reserved.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®
CICS®
Database 2™
DB2®

IBM®
ibm.com®
IMS™
Lotus®
Notes®
PureCoverage®
Purify®
Quantify®
Rational®
Redbooks™
Redbooks (logo)
Tivoli Enterprise™
Tivoli Enterprise Console®
Tivoli Management Environment®
Tivoli®
TME®
WebSphere®

The following terms are trademarks of other companies:

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM® Redbook will help you install, tailor, and configure the new IBM Tivoli Monitoring for Transaction Performance Version 5.2, which will assist you in determining the business performance of your e-business transactions in terms of responsiveness, performance, and availability.

The major enhancement in Version 5.2 is the addition of state-of-the-art industry strength monitoring functions for J2EE applications hosted by WebSphere® Application Server or BEA Weblogic. In addition, the architecture of Web Transaction Monitoring (WTP) has been redesigned to provide for even easier deployment, increased scalability, and better performance. Also, the reporting functions has been enhanced by the addition of ETL2s for the Tivoli Enterprise Date Warehouse.

This new version of IBM Tivoli® Monitoring for Transaction Performance provides all the capabilities of previous versions of IBM Tivoli Monitoring for Transaction Performance, including the Enterprise Transaction Performance (ETP) functions used to add transaction performance monitoring capabilities to the Tivoli Management Environment® (with the exception of reporting through Tivoli Decision Support). The reporting functions have been migrated to the Tivoli Enterprise Date Warehouse environment.

Because the ETP functions has been documented in detail in the redbook *Unveil Your e-business Transaction Performance with IBM TMTP 5.1,* SG24-6912, this publication is devoted to the Web Transaction Performance functions of IBM Tivoli Monitoring for Transaction Performance Version 5.2, and, in particular, the J2EE monitoring capabilities.

This information in this redbook is organized in three major parts, each targeted at specific audiences:

Part 1, “Business value of end-to-end transaction monitoring” on page 1 provides a general overview of IBM Tivoli Monitoring for Transaction Performance and discusses the transaction monitoring needs of an e-business, in particular, the need for monitoring J2EE based applications. The target audience for this section is decision makers and others that need a general understanding of the capabilities of IBM Tivoli Monitoring for Transaction Performance and the challenges, from a business perspective, that the product helps address. This section is organized as follows:

- Chapter 1, “Transaction management imperatives” on page 3

© Copyright IBM Corp. 2003. All rights reserved.

- Chapter 2, “IBM Tivoli Monitoring for Transaction Performance in brief” on page 37
- Chapter 3, “IBM TMTP architecture” on page 55

Part 2, “Installation and deployment” on page 83 is targeted towards persons that are interested in implementing issues regarding IBM Tivoli Monitoring for Transaction Performance. In this section, we will describe best practices for installing and deploying the Web Transaction Performance components of IBM Tivoli Monitoring for Transaction Performance Version 5.2, and we provide information on how to ensure the operation of the tool. This section includes:

- Chapter 4, “TMTP WTP Version 5.2 installation and deployment” on page 85
- Chapter 5, “Interfaces to other management tools” on page 153
- Chapter 6, “Keeping the transaction monitoring environment fit” on page 177

Part 3, “Using TMTP to measure transaction performance” on page 209 is aimed at the audience that will use IBM Tivoli Monitoring for Transaction Performance functions on a daily basis. Here, we provide detailed information and best practices on how to configure monitoring policies and deploy monitors to gather transaction performance data. We also provide extensive information on how to create meaningful reports from the data gathered by IBM Tivoli Monitoring for Transaction Performance. This part includes:

- Chapter 7, “Real-time reporting” on page 211
- Chapter 8, “Measuring e-business transaction response times” on page 225
- Chapter 9, “Rational Robot and GenWin” on page 325
- Chapter 10, “Historical reporting” on page 375

It is our hope that this redbook will help you enhance your e-business management solutions to benefit your organization and better support future Web based initiatives.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.

Morten Moeller is an IBM Certified IT Specialist working as a Project Leader at the International Technical Support Organization, Austin Center. He applies his extensive field experience as an IBM Certified IT Specialist to his work at the ITSO where he writes extensively on all areas of Systems Management. Before joining the ITSO, Morten worked in the Professional Services Organization of IBM Denmark as a Distributed Systems Management Specialist, where he was

involved in numerous projects designing and implementing systems management solutions for major customers of IBM Denmark.

Sanver Ceylan is an Associate Project Leader at the International Technical Support Organization, Austin Center. Before working with the ITSO, Sanver worked in the Software Organization of IBM Turkey as an Advisory IT Specialist, where he was involved in numerous pre-sales projects for major customers of IBM Turkey. Sanver holds a Bachelors degree in Engineering Physics and a Masters degree in Computer Science.

Mahfujur Bhuiyan is a Systems Specialist and Certified Tivoli Enterprise™ Consultant at TeliaSonea IT-Service, Sweden. Mahfujur has over eight years of experience in Information Technology with a focus on systems and network management and distributed environment, and was involved in several projects in designing and implementing Tivoli environments for TeliaSonena's external and internal customers. He holds a Bachelors degree in Mechanical Engineering and a Masters degree in Environmental Engineering from the Royal Institute of Technology (KTH), Sweden.

Valerio Graziani is a Staff Engineer at the IBM Tivoli Laboratory in Italy with nine years of experience in software development and verification. He currently leads the System Verification Test on IBM Tivoli Monitoring. He has been an IBM employee since 1999 after working as an independent consultant for large software companies since 1994. He has three years of experience in the application performance measurement field. His areas of expertise include test automation, performance and availability monitoring, and systems management.

Scott Henley is an IBM System Engineer based in Australia who performs pre and post-sales support for IBM Tivoli products. Scott has almost 15 years of Information Technology experience with a focus on Systems Management utilizing IBM Tivoli products. He holds a Bachelors degree in Information Technology from Australia's Charles Stuart University and is due to complete his Masters in Information Technology in 2004. Scott holds product certifications for many of IBM Tivoli PACO and Security products, as well as MCSE status since 1997 and the RHCE status since 2000.

Zoltan Veress is an independent System Management Consultant working for IBM Global Services, France. He has eight years of experience in the field. His major areas of expertise include software distribution, inventory, remote control, and he also has experience with almost all Tivoli Framework-based products.

Thanks to the following people for their contributions to this project:

The Editing Team
International Technical Support Organization, Austin Center

Fergus Stewart, Randy Scott, Cheryl Thrailkill, Phil Buckellew, David Hobbs
Tivoli Product Management

Russ Blaisdell, Oliver Hsu, Jose Nativio, Steven Stites, Bret Patterson, Mike Kiser, Nduwuisi Emuchay
Tivoli Development

J.J. Garcia, Greg K Havens II, Tina Lamacchia
Tivoli SWAT Team

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

Part 1

Business value of end-to-end transaction monitoring

In this part, we discuss an overview of transaction management imperatives, a high-level brief of IBM Tivoli Monitoring for Transaction Performance 5.2, and a high-level and detailed architectural concept.

© Copyright IBM Corp. 2003. All rights reserved.

The following main topics are included:

- Chapter 1, “Transaction management imperatives” on page 3
- Chapter 2, “IBM Tivoli Monitoring for Transaction Performance in brief” on page 37
- Chapter 3, “IBM TMTP architecture” on page 55

1

Transaction management imperatives

This chapter provides an overview of the business imperatives for looking at transaction performance. We also use this chapter to discuss, in broader terms, the topics of system management and availability, as well as performance monitoring.

© Copyright IBM Corp. 2003. All rights reserved.

1.1 e-business transactions

In the Web world, users perceive interacting with an organization or a business through a Web-based interface as a single, continuous interaction or session between the user's machine and the systems of the other party, and that is how it should be. However, the interaction is most likely made up of a large number of individual, interrelated transactions, each one providing its own specific part of the complex set of functions that implement an e-business transaction, perhaps running on systems owned by other organizations or legal entities.

Figure 1-1 shows a typical Web-based transaction, the resources used to facilitate the transaction, and the typical components of a transaction breakdown.

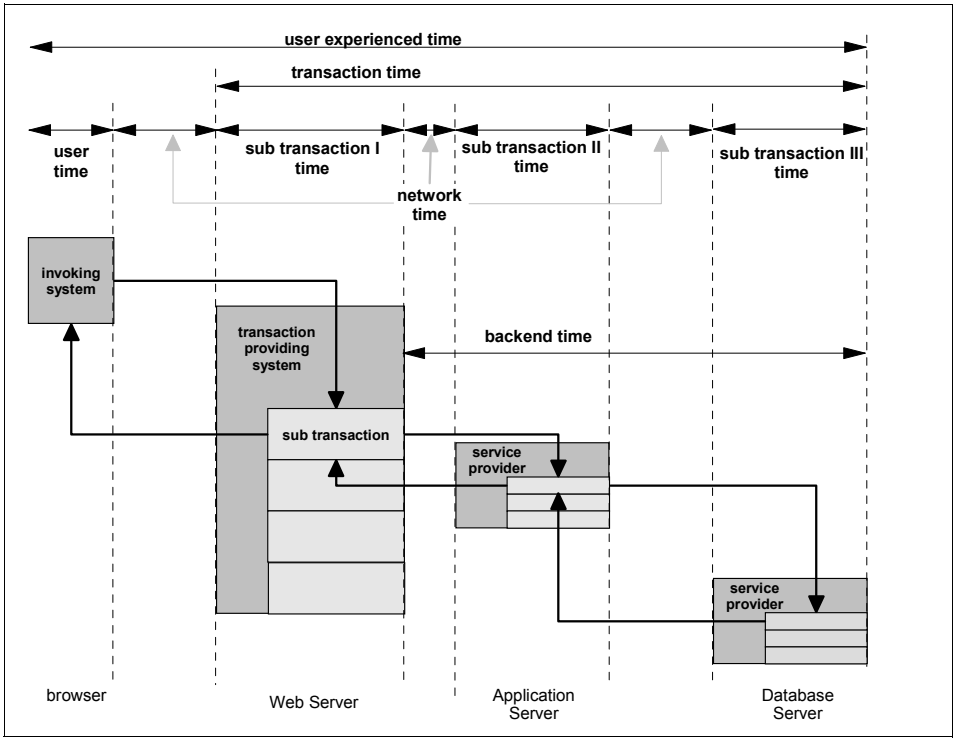


Figure 1-1 Transaction breakdown

In the context of this book, we will differentiate between different types of transactions depending on the location of the machine from which the transaction is initiated:

Web transaction Originate from the Internet, thus we have no predetermined knowledge about the user, the system, and the location of the transaction originator.

Enterprise transaction Initiated from well-known systems, most of which are under our control, and knowledge of the available resources exists. Typically, the systems initiating these types of transactions are managed by our Tivoli Management Environment.

Application transaction Subtransactions that are initiated by the application-provisioning Web transactions to the end users. Application transactions are typically, but not always, also enterprise transactions, but also may initiate from third-party application servers.

A typical application transaction is a database lookup performed from a Web application server, in response to a Web transaction initiated by an end user.

From a management point of view these transaction types should be treated similarly. Responsiveness from the Web application servers to any requester is equally important, and it should not make a difference if the transaction has been initiated from a Web user, an internal user, or a third-party application server. However, business priorities may influence the level of service or importance given to individual requestors.

However, it is important to note that monitoring transaction performance does not in any way obviate the need to perform the more traditional systems management disciplines, such as capacity, availability, and performance management. Since the Web applications are comprised of several resources, each hosted by a server, these individual server resources must be managed to ensure that they provide the services required by the applications.

With the myriad servers (and exponentially more individual resources and components) involved in an average-sized Web application system, management of all of these resources is more an art than a science. We begin by providing a short description of the challenges of e-business provisioning in order to identify the management needs and issues related to provisioning e-business applications.

1.2 J2EE applications management

Application management is one of the fastest growing areas of infrastructure management. This is a consequence of the focus on user productivity and confirms the fact that more and more we are moving away from device-centric management. Within this segment today, J2EE platform management is only a fairly small component. However, it is easy to foresee that J2EE is one of the

next big things in application architecture, and because of this, we may well see this area converted into a bigger slice of the pie, and eventually envision much of the application management segment being dedicated to J2EE.

Because J2EE based applications cover multiple internal and external components, they are more closely tied to the actual business process than other types of application integration schemes used before. The direct consequence of this link between business process and application is that management of these application platforms must provide value in several dimensions, each targeted to a specific constituency within the enterprise, such as:

- The enterprise groups interested in the different phases of a business process and in its successful completion
- The application groups with an interest in the quality of the different logical components of the global application
- The IT operations group providing infrastructure service assurance and interested in monitoring and maintaining the services through the application and its supporting infrastructure

People looking for a J2EE management solution must make sure that any product they select does, along with other enterprise-specific requirements, provide the data suited to these multiple reporting needs.

Application management represents around 24% of the infrastructure performance management market. But the new application architecture enabled by J2EE goes beyond application management. The introduction of this new application architecture has the potential not only to impact the application management market, but also, directly or indirectly, to disrupt the whole infrastructure performance market by forcing a change in the way enterprises implement infrastructure management. The role of J2EE application architectures goes beyond a simple alternative to traditional transactional application. It has the potential to link applications and services residing on multiple platforms, external or internal, in a static or dynamic, loosely coupled relationship that models a business process much more closely than any other application did. It is also a non-device platform, yet it is an infrastructure component with the usual attributes of a hard component in terms of configuration and administration. But its performance is also related and very dependent on the resources of supporting components, such as servers, networks, and databases. The consequences of this profound modification in application architecture will ripple, over time, into the way the supporting infrastructure is managed.

The majority of today's infrastructure management implementations are confined to devices monitored in real time for fault and performance from a central enterprise console.

In this context, application management is based on a traditional agent-server relationship, collecting data mostly from the outside, with little insight into the application internals. For example:

- Standard applications may provide specific parameters (usually resource consumption) to a custom agent.
- Custom applications are mostly managed from the outside by looking at their resource consumption.

In-depth analysis of application performance using this approach is not a real-time activity, and the most common way to manage real-time availability and performance (response time) of applications is to use external active agents.

Service-level management, capacity planning, and performance management are aimed at the devices and remain mostly “stove-piped” activities, essentially due to the inability of the solutions used to automatically model the infrastructure supporting an application or a business process.

This proved to be a problem already in client/server implementations, where applications spanned multiple infrastructure components. This problem is magnified in J2EE implementations.

1.2.1 The impact of J2EE on infrastructure management

J2EE architecture brings important changes to the way an application is supported by the underlying infrastructure. In the distributed environment, a direct relationship is often believed to exist between the hardware resources and the application performance. Consequently, managing the hardware resources by type (network, servers, and storage) is often thought to be sufficient.

J2EE infrastructure does not provide this one-to-one relation between application and hardware resource. The parameters driving the box performances may reflect the resource usage of the Java™ Virtual Machine (JVM), but they cannot be associated directly with the performance of the application, which may be driven either by its own configuration parameters within the JVM, or by the impact of external component performances.

The immediate consequence on infrastructure management is that a specific monitoring tool has to be included in the infrastructure management solution to care for the specificities of the J2EE application server, and that the application has to be considered as a service spanning multiple components (a typical J2EE application architecture is described in 3.6, “Putting it all together” on page 80), where the determination of a problem’s origin requires some intelligence based on predefined rules or correlation. This requires expertise in the way the

application is designed and the ability to include this expertise in the problem resolution process.

Another set of problems is posed by the ability to federate multiple applications from the J2EE platform using Enterprise Application Integration (EAI) to connect to existing applications, the generation of complementary transactions with external systems, or the inclusion of Web Services. This capability brings the application closer to the business process than before since multiple steps, or phases, of the process, which were performed by separate applications, are now integrated. The use of discrete steps in a business process allowed for a manual check on their completion, a control that is no longer available in the integrated environment and must be replaced by data coming from infrastructure management. This has consequences not only on where the data should be captured, but also on the nature of the data itself. Finally, the complexity of the application created by assembling diverse components makes quality assurance (QA) a task that is both more important than ever and almost impossible to complete with the degree of certainty that was available in other applications. Duplicating the production environment in a test environment becomes difficult. To be more effective, operations should participate in QA to bring infrastructure expertise into the process and should also be prepared to use QA as a resource during operations to test limited changes or component evolution.

The infrastructure management solution adapted to the new application architecture must include a real-time monitoring component that provides a “service assurance” capability. It must extend its data capture to all components, including J2EE and connectors, to other resources, such as EAI, and be able to collect additional parameters beyond availability and performance. Content verification and security are some of the possible parameters, but “transaction availability” is another type of alert that becomes relevant in this context close to the business process.

Root-cause analysis, which identifies the origin of a problem in real time, must be able to pinpoint problems within the transaction flow, including the J2EE application server and the external components of the application.

An analytical component, to help analyze problems within and without the application server, is necessary to complement the more traditional tools aimed at analyzing infrastructure resources.

1.2.2 Importance of JMX

In the management of J2EE platforms, the JMX model has emerged as an important step in finding an adaptable management model.

The Java Management Extensions (JMX) technology represents a universal, open technology for management and monitoring that can be deployed wherever management and monitoring are needed. JMX is designed to be suitable for adapting legacy systems, implementing new management and monitoring solutions, and plugging into future monitoring systems.

JMX allows centralized management of managed beans, or MBeans, which act as wrappers for applications, components, or resources in a distributed network. This functionality is provided by a MBean server, which serves as a registry for all MBeans, exposing interfaces for manipulating them. In addition, JMX contains the m-let service, which allows dynamic loading of MBeans over the network. In the JMX architectural model, the MBean server becomes the spine of the server where all server components plug in and discover other MBeans via the MBean server notification mechanism.

The MBean server itself is extremely lightweight. Thus, even some of the most fundamental pieces of the server infrastructure are modeled as MBeans and plugged into the MBean server core, for example, protocol adapters. Implemented as MBeans, they are capable of receiving requests across the network from clients operating in different network protocols, like SNMP and WBEM, enabling JMX-based servers to be managed with tools written in any programming language. The result is an extremely modular server architecture, and a server easily managed and configured remotely using a number of different types of tools.

Impact on IT organizations

The addition of tools requires adequate training in their use. But the types of problems that these tools are going to uncover also require skills and organizational groups with IT operations. For example:

- The capability to handle more event types in the operation center. Transaction availability events and performance events are typical of the new applications. This requires that the operation center understand the impact of these events and the immediate action required to maintain the service in a service assurance-oriented, rather than “network and system management”-oriented, environment.
- The capability to handle and analyze application problems, or what appears to be application problems. This requires that the competency groups in charge of finding permanent “fixes” understand the application architecture and are able to address the problems.
- A stronger cooperation between QA and operations to make sure that the testing phase is a true preparation of the deployment phase, and that recurring tests are made following changes and fixes. Periodic tests to validate performance and capacity parameters are also good practice.

While service assurance and real-time root-cause analysis are attractive propositions, the J2EE management market is not yet fully mature. Combined with the current economic climate, this means that a number of the solutions available today may disappear or be consolidated within stronger competitors tomorrow. Beyond a selection based on pure technology and functional merits, clients should consider the long-term viability of the vendor before making a decision that will have such an impact on their infrastructure management strategies.

J2EE application architectures have, and will continue to have, a strong impact on managing the enterprise infrastructure. As the future application model is based on a notion of service rather than a suite of discrete applications, the future model of infrastructure management will be based on service assurance rather than event management. An expanded set of parameters and a close integration within a real-time operational model offering root-cause analysis is necessary.

Recommendations

The introduction of J2EE application servers in the enterprise infrastructure is having a profound impact on the way this infrastructure is managed. Potential availability, performance, quality, and security problems will be magnified by the capabilities of the application technology, with consequences in the way problems are identified, reported, and corrected. As J2EE technologies become mainstream, the existing infrastructure management processes, which are focused today mostly on availability and performance, will have to evolve toward service assurance and business systems management. Organizations should look at the following before selecting a tool for transaction monitoring:

1. The product selected for the management of the J2EE application server meets the following requirements:
   a. Provides a real-time (service assurance) and an in-depth analysis component, preferably with a root-cause analysis and corrective action mechanism.
   b. Integrates with the existing infrastructure products, downstream (enterprise console and help desk) and upstream (reuse of agents).
   c. Provides customized reporting for the different constituencies (business, development, and operations).
2. The IT operation organization is changed (to reflect the added complexity of the new application infrastructure) to:
   a. Handle more event types in the operation center. Transaction availability events and performance events are typical of the new applications as well as events related to configuration and code problems.

b. Create additional competency groups within IT operation, with the ability to receive and analyze application-related problems in cooperation with the development groups.

c. Improve the communication and cooperation between competency silos within IT operations, since many problems are going to involve multiple hardware and software platforms.

d. Establish or improve the cooperation between QA and operations to make sure that the testing phase is a true preparation of the deployment phase, and that many integration and performance problems are tackled beforehand.

1.3 e-business applications: complex layers of services

A modern e-business solution is much more complex than the standard terminal processing-oriented systems of the 1970s and 1980s, as illustrated in Figure 1-2 on page 12. However, despite major revisions, especially during the turn of the last century, legacy systems are still the bread-and-butter of many enterprises, and the e-business solutions in these environments are designed to front-end these mainframe-oriented application complexes.

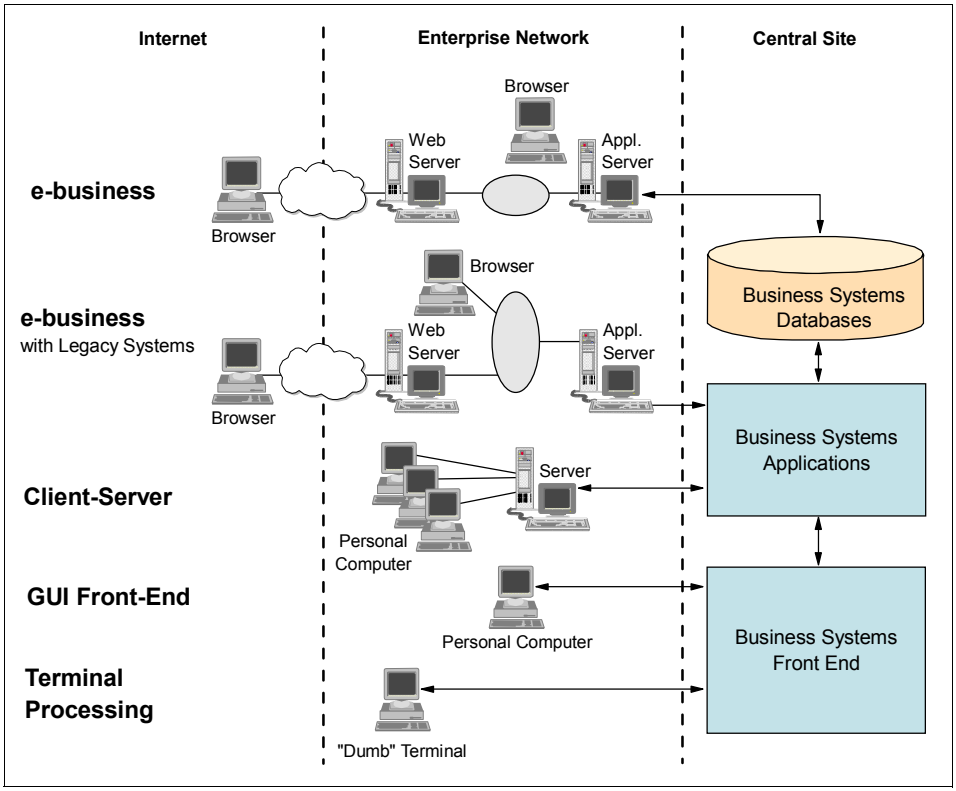


Figure 1-2 Growing infrastructure complexity

The complex infrastructure needed to facilitate e-business solutions has been dictated mostly by requirements for standardization of client run-time environments in order to allow any standard browser to access the e-business sites. In addition, application run-time technologies play a major role, as they must ensure platform independence and seamless integration to the legacy back-end systems, either directly to the mainframe or through the server part of the old client-server solution. Furthermore, making the applications accessible from anywhere in the world by any person on the planet raises some security issues (authentication, authorization, and integrity) that did not need addressing in the old client-server systems, as all clients were well-known entities in the internal company network.

Because of the central role that the Web and application servers play within a business and the fact that they are supported and typically deployed across a

variety of platforms throughout the enterprise, there are several major challenges to managing the e-business infrastructure, including:

- Managing Web and application servers on multiple platforms in a consistent manner from a central console
- Defining the e-business infrastructure from one central console
- Monitoring Web resources (sites and applications) to know when problems have occurred or are about to occur
- Taking corrective actions when a problem is detected in a platform independent way
- Gathering data across all e-business environments to analyze events, messages, and metrics

The degree of complexity of e-business infrastructure system management is directly proportional to the size of the infrastructure being managed. In its simplest form, an e-business infrastructure is comprised of a single Web server and its resources, but it can grow to hundreds or even thousands of Web and application servers throughout the enterprise.

To add to the complexity, the e-business infrastructure may span many platforms with different network protocols, hardware, operating systems, and applications. Each platform possesses its unique and specific systems management needs and requirements, not to mention a varying level of support for the administrative tools and interfaces.

Every component in the e-business infrastructure is a potential show-stopper, bottleneck or even single point of failure. Each and every one provides specialized services needed to facilitate the e-business application system. The term *application systems* is used deliberately to enforce the point that no single component by itself provides a total solution: the application is pieced together by a combination of standard off-the-shelf components and home-grown components. The standard components provide general services, such as session control, authentication and access control, messaging, and database access, and the home-grown components add the application logic needed to glue all the different bits and pieces together to perform the specific functions for that application system. On an enterprise level, chances are that many of the home-grown components may be promoted to standard status to ensure specific company standards or policies.

At first glance, breaking up the e-business application into many specialized services may be regarded as counterproductive and very expensive to implement. However, specialization enables sharing of common components (such as Web, application, security, and database servers) between more e-business application systems, and it is key to ensuring availability and

performance of the application system as a whole by allowing for duplication and distribution of selected components to meet specific resource requirements or increase the performance of the application systems as a whole. In addition, this itemizing of the total solution allows for almost seamless adoption of new technologies for selected areas without exposing the total system to change.

Whether the components in the e-business system are commercial, standard, or application-specific, each of them will most likely require other general services, such as communication facilities, storage space, and processing power, and the computers on which they run need electrical power, shelter from rain and sun, access security, and perhaps even cooling.

As it turns out, the e-business application relies on several layers of services that may be provided internally or by external companies. This is illustrated in Figure 1-3.

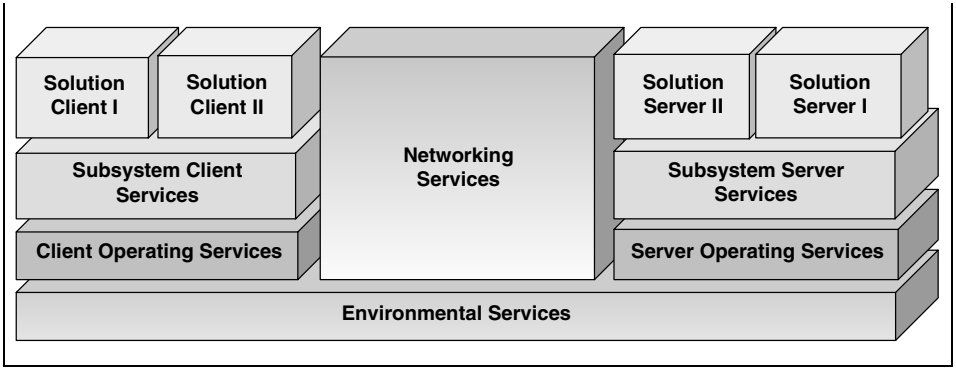


Figure 1-3 Layers of service

As a matter of fact, it is not exactly the e-business application that relies on the services depicted above. The correct notion is that individual components (such as Web servers, database servers, application servers, lines, routers, hubs, and switches) each rely on underlying services provided by some other component. This can be broken down even further, but that is beyond this discussion. The point is that the e-business solution is exactly as solid, robust, and stable as the weakest link of the chain of services that make up the entire solution, and since the bottom-line results of an enterprise may be affected drastically by the quality of the e-business solutions provided, a worst-case scenario may prove that a power failure in Hong Kong may have an impact on sales figures in Greece and that increased surface activity on the sun may result in satellite-communication problems that prevent car rental in Chattanooga.

While mankind cannot prevent increased activity of the sun and wind, there are a number of technologies available to allow for continuing, centralized monitoring

and surveillance of the e-business solution components. These technologies will help manage the IT resources that are part of the e-business solution. Some of these technologies may even be applied to manage the non-IT resources, such as power, cooling, and access control.

However, each layer in any component is specialized and requires different types of management. In addition, from a management point of view, the top layer of any component is the most interesting, as it is the layer that provides the unique service that is required by that particular component. For a Web server, the top layer is the HTTP server itself. This is the mission-critical layer, even though it still needs networking, an operating system, hardware, and power to operate. On the other hand, for an e-business application server (although it also may have a Web server installed for communicating with the dedicated Web Server), the mission-critical layer is the application server, and the Web server is considered secondary in this case, just as the operating system, power, and networking are. This said, all the underlying services are needed and must operate flawlessly in order for the top layer to provide its services. It is much like driving a car: you monitor the speedometer regularly to avoid penalties by violating changing speed limits, but you check the fuel indicator only from time to time or when the indicator alerts you to perform preventive maintenance by filling up the tank.

1.3.1 Managing the e-business applications

Specialized functions require specialized management, and general functions require general management. Therefore, it is obvious that the management of the operating system, hardware layer, and networking layer may be may be general, since they are used by most of the components of the e-business infrastructure. On the other hand, a management tool for Web application servers might not be very well-suited for managing the database server.

Up till now, the term “managing” has been widely used, but not yet explained. Control over and management of the computer system and its vital components are critical to the continuing operation of the system and therefore the timely availability of the services and functions provided by the system. This includes controlling both physical and logical access to the system to prevent unauthorized modifications to the core components, and monitoring the availability of the systems as a whole, as well as the performance and capacity usage of the individual resources, such as disk space, networking equipment, memory, and processor usage. Of course, these control and monitoring activities have to be performed cost-effectively, so the cost of controlling any resource does not become higher than the cost of the resource itself. It does not make much business sense to spend $1000 to manage a $200 hard disk, unless the data on that hard disk represents real value to the business in excess of $1000. Planning for recovery of the systems in case of a disaster also needs to be

addressed, as being without computer systems for days or weeks may have a huge impact on the ability to conduct business.

There still is one important aspect to be covered for successfully managing and controlling computer systems. We have mentioned various hardware and software components that collectively provide a service, but which components are part of the IT infrastructure, where are they, and how do they relate to one another? A prerequisite for successful management is the detailed knowledge of which components to manage, how the components interrelate, and how these components may be manipulated in order to control their behavior.

In addition, now that IT has become an integral part of doing business, it is equally important from an IT management point of view to know which commitments we have made with respect to availability and performance of the e-business solutions, and what commitments our subcontractors have made to us. And for planning and prioritization purposes, it is vital to combine our knowledge about the components in the infrastructure with the commitments we have made in order to assess and manage the impact of component malfunction or resource shortage. In short, in a modern e-business environment, one of the most important management tasks is to control and manage the service catalogue in which all the provisioned services are defined and described, and the SLAs in which the commitments of the IT department are spelled out.

For this discussion, we turn to the widely recognized Information Technology Infrastructure Library (ITIL). The ITIL was developed by the British Government's Central Computer and Telecommunications Agency (CCTA), but has over the past decade or more gained acceptance in the private sector.

One of the reasons behind this acceptance is that most IT organizations, met with requirements to promise or even guarantee performance and availability, agree that there is no point in agreeing to deliver a service at a specific level if the basic tools and processes needed to deploy, manage, monitor, correct, and report the achieved service level have not been established. ITIL groups all of these activities into two major areas, Service Delivery and Service Support, as shown in Figure 1-4 on page 17.

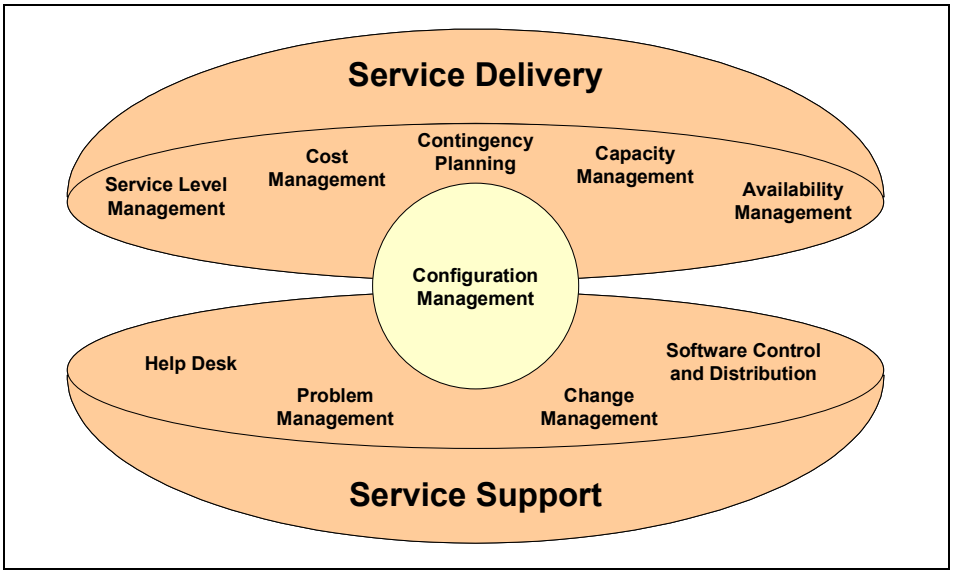


Figure 1-4 The ITIL Service Management disciplines

The primary objectives of the Service Delivery discipline are proactive and consist primarily of planning and ensuring that the service is delivered according to the Service Level Agreement. For this to happen, the following tasks have to be accomplished.

Service Delivery

Within ITIL, the proactive disciplines are grouped in the Service Delivery area, which are covered in the following sections.

Service Level Management

Service Level Management involves managing customer expectations and negotiating Service Level Agreements. This involves identifying customer requirements and determining how these can best be met within the agreed-upon budget, as well as working together with all IT disciplines and departments to plan and ensure delivery of services. This involves setting measurable performance targets, monitoring performance, and taking action when targets are not met.

Cost Management

Cost Management consists of registering and maintaining cost accounts related to the use of IT services and delivering cost statistics and reports to Service Level Management to assist in obtaining the correct balance between service

cost and delivery. It also means assisting in pricing the services in the service catalog and SLAs.

Contingency Planning

Contingency Planning develops and ensures the continuing delivery of minimum outage of the service by reducing the impact of disasters, emergencies, and major incidents. This work is done in close collaboration with the company's business continuity management, which is responsible for protecting all aspects of the company's business, including IT.

Capacity Management

Capacity Management plans and ensures that adequate capacity with the expected performance characteristics is available to support the service delivery. It also delivers capacity usage, performance, and workload management statistics (as well as trend analysis) to Service Level Management.

Availability Management

Availability Management means planning and ensuring the overall availability of the services and providing management information in the form of availability statistics, including security violations, to Service Level Management.

Even though not explicitly mentioned in the ITIL definition, for this discussion, content management is included in this discipline.

This discipline may also include negotiating underpinning contracts with external suppliers and the definition of maintenance windows and recovery times.

The disciplines in the Service Support group are mainly reactive and are concerned with implementing the plans and providing management information regarding the levels of service achieved.

Service Support

The reactive disciplines that are considered part of the Service Support group are shown in the following sections.

Configuration Management

Configuration Management is responsible for registering all components in the IT service, including customers, contracts, SLAs, hardware and software components, and maintaining a repository of configured attributes and relationships between the components.

Help Desk

The Help Desk acts as the main point of contact for users of the service. It registers incidents, allocates severity, and coordinates the efforts of support teams to ensure timely and accurate problem resolution.

Escalation times are noted in the SLA and are agreed on between the customer and the IT department. The Help Desk also provides statistics to Service Level Management to demonstrate the service levels achieved.

Problem Management

Problem Management implements and uses procedures to perform problem diagnosis and identify solutions that correct problems. It also registers solutions in the configuration repository.

Escalation times should be agreed upon internally with Service Level Management during the SLA negotiation. It also provides problem resolution statistics to support Service Level Management.

Change Management

Change Management plans and ensures that the impact of a change to any component of a service is well known and that the implications regarding service level achievements are minimized. This includes changes to the SLA documents and the Service Catalog as well as organizational changes and changes to hardware and software components.

Software Control and Distribution

It is the responsibility of Software Control and Distribution to manage the master software repository and deploy software components of services. It also deploys changes at the request of Change Management, and provides management reports regarding deployment.

The key relationships between the disciplines are shown in Figure 1-5 on page 20.

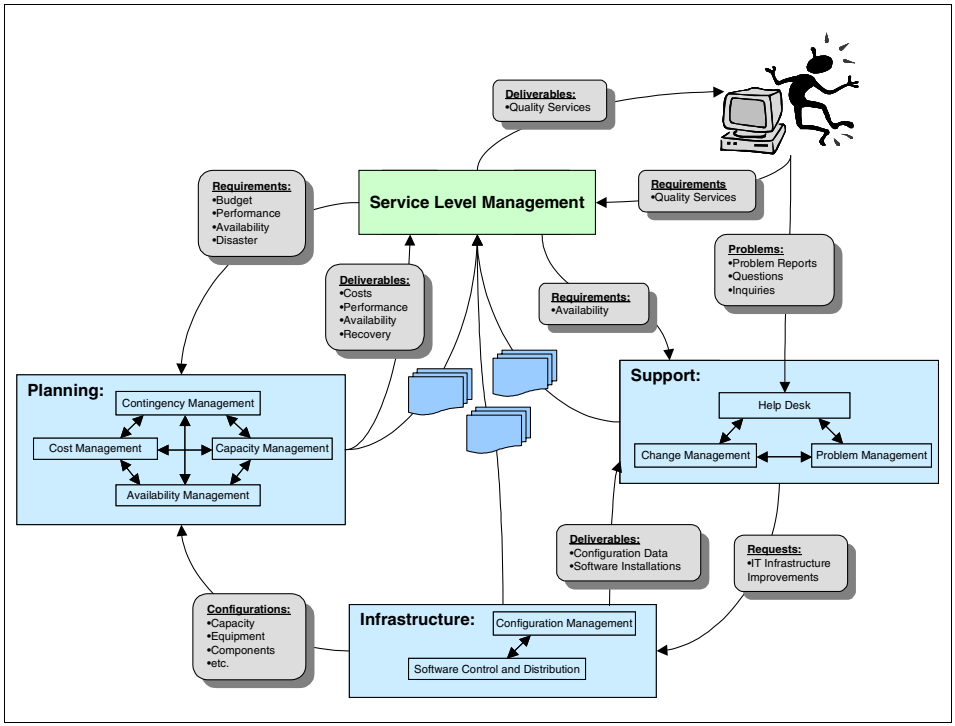


Figure 1-5 Key relationships between Service Management disciplines

For the remainder of this discussion, we will limit our discussion to capacity and availability management of the e-business solutions. Contrary to the other disciplines that are considered common for all types of services provided by the IT organization, the e-business solutions provide special challenges to management, due to their high visibility and importance to the bottom line business results, their level of distribution, and the special security issues that characterize the Internet.

1.3.2 Architecting e-business application infrastructures

In a typical e-business environment, the application infrastructure consists of three separate tiers, and the communication between these is restricted, as Figure 1-6 shows.

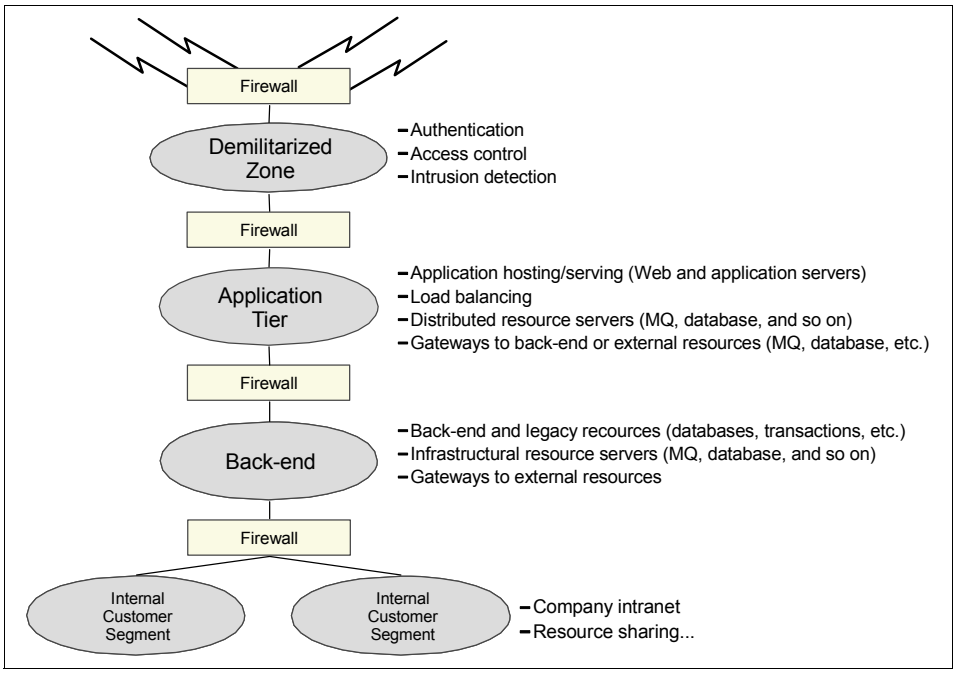


Figure 1-6 A typical e-business application infrastructure

The tiers are typically:

Demilitarized Zone The tier accessible by all external users of the applications. This tier functions as the gatekeeper to the entire system, and functions such as access control and intrusion detection are enforced here. The only other part of the intra-company network that the DMZ can talk to is the application tier.

Application Tier This is usually implemented as a dedicated part of the network where the application servers reside. End-user requests are routed from the DMZ to the specific servers in this tier, where they are serviced. In case the applications need to use resources from company-wide databases, for example, these are requested from the back-end tier, where all the secured company IT assets reside. As was the case for communication between the DMZ and the

Application Tier, the communication between the Application Tier and the back-end systems is established through firewalls and using well-known connection ports. This helps ensure that only known transactions from known machines outside the network can communicate with the company databases or legacy transaction systems (such as CICS® or IMS™). Apart from specific application servers, this tier also hosts load-balancing devices and other infrastructural components (such as MQ Servers) needed to implement a given application architecture.

Back-end Tier This is where all the vital company resources and IT assets reside. External access to these resources is only possible through the DMZ and the Application Tier.

This model architecture is a proven way to provide secure, scalable, high-availability external access to company data with a minimum of exposure to security violations. However, the actual components, such as application servers and infrastructural resources, may vary depending upon the nature of the applications, company policies, the requirements to availability and performance, and the capabilities of the technologies used.

If you are in the e-business hosting area or you have to support multiple lines of business that require strict separation, the conceptual architecture shown in Figure 1-6 on page 21 may be even more complicated. In these situations, one or more of the tiers may have to be duplicated to provide the required separation. In addition, the back-end tier might even be established remotely (relative to the application tier). This is very common when the e-business application hosting is outsourced to an external vendor, such as IBM Global Services.

To help design the most appropriate architecture for a specific set of e-business applications, IBM has published a set of e-business patterns that may be used to speed up the process of developing e-business applications and deploying the infrastructure to host them.

The concept behind these e-business patterns is to reuse tested and proven architectures with as little modification as possible. IBM has gathered experiences from more than 20,000 engagements, compiled these into a set of guidelines, and associated them with links. A solution architect can start with a problem and a vision for the solution and then find a pattern that fits that vision. Then, by drilling down using the patterns process, the architect can further define the additional functional pieces that the application will need to succeed. Finally, the architect can build the application using coding techniques outlined in the associated guidelines. Further details on e-business patterns may be found in Appendix A, “Patterns for e-business” on page 429.

For a full understanding of the patterns, please review the book *Patterns for e-business: A Strategy for Reuse* by Adams, et al.

1.3.3 Basic products used to facilitate e-business applications

So far, we may conclude that building an e-business solution is like building a vehicle, in the sense that:

- We want to provide the user with a standard, easy-to-use interface that fulfills the needs of the user and has a common look-and-feel to it.
- We want to use as many standard components as possible to keep costs down and be able to interchange them seamlessly.
- We want it to be reliable and available at all times with a minimum of maintenance.
- We want to build in unique features (differentiators) that make the user choose our product over those of the competitors.

The main difference between the vehicle and the e-business solution is that we own and control the solution, but the buyer owns and manages the vehicle. The vehicle owner decides when to have the oil changed and when to fill up the fuel tank or adjust the tire pressure. The vehicle owner also decides when to take the vehicle in for a tune-up, when to add chrome bumpers and alloy wheels to make the vehicle look better, and when to sell it. The user of an e-business site has none of those choices. As owners of the e-business solution, we decide when to rework the user interface to make it look better, when to add resources to increase performance, and ultimately when to retire and replace the solution.

This gives us a few advantages over the car manufacturer, as we can modify the product seamlessly by adding or removing components as needed in order to align the performance with the requirements and adjust the functionality of the product as competition toughens or we engage in new alliances.

No matter whether the e-business solution is the front-end of a legacy system or a new application developed using modern, state-of-the-art development tools, it may be characterized by three specific layers of services that work together to provide the unique functionality necessary to allow the applications to be used in an Internet environment, as shown in Figure 1-7 on page 24.

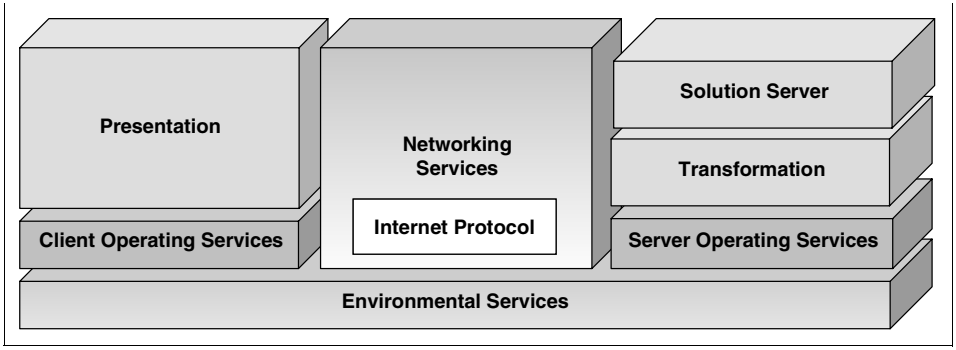


Figure 1-7 e-business solution-specific service layers

The presentation layer must be a commonly available tool that is installed on all the machines used by users of the e-business solution. It should support modern development technologies such as XML, JavaScript, and HTML pages, and usually is the browser.

The standard communication protocols used to provide connectivity using the Internet are TCP/IP, HTTP, and HTTPS. These protocols must be supported by both client and server machines.

The transformation services are responsible for receiving client requests and transforming them into business transactions that in turn are served by the Solution Server. In addition, it is the responsibility of the transformation service to receive results from the Solution Server and convey them back to the client in a format that can be handled by the browser. In e-business solutions that do not interact with legacy systems, the transformation and Solution Server services may be implemented in the same application, but most likely they are split into two or more dedicated services.

This is a very simple representation of the functions that take place in the transformation service. Among other functions that must be performed are identification, authentication and authorization control, load balancing, and transaction control. Dedicated servers for each of these functions are usually implemented to provide a robust and scalable e-business environment. In addition, some of these are placed in a dedicated network segment (the demilitarized zone (DMZ)), which, from the point of view of the e-business owner, is fully controlled, and in which client requests are received by “well-known,” secure systems and passed on to the enterprise network, also known as the intranet. This architecture is used to increase security by avoiding transactions from “unknown” machines to reach the enterprise network, thereby minimizing the exposure of enterprise data and the risk of hacking.

To facilitate secure communication between the DMZ and the intranet, a set of Web servers is usually implemented, and identification, authentication, and authorization are typically handled by an LDAP Server.

The infrastructure depicted in Figure 1-8 contains all components required to implement a secure e-business solution, allowing anyone from anywhere to access and do business with the enterprise.

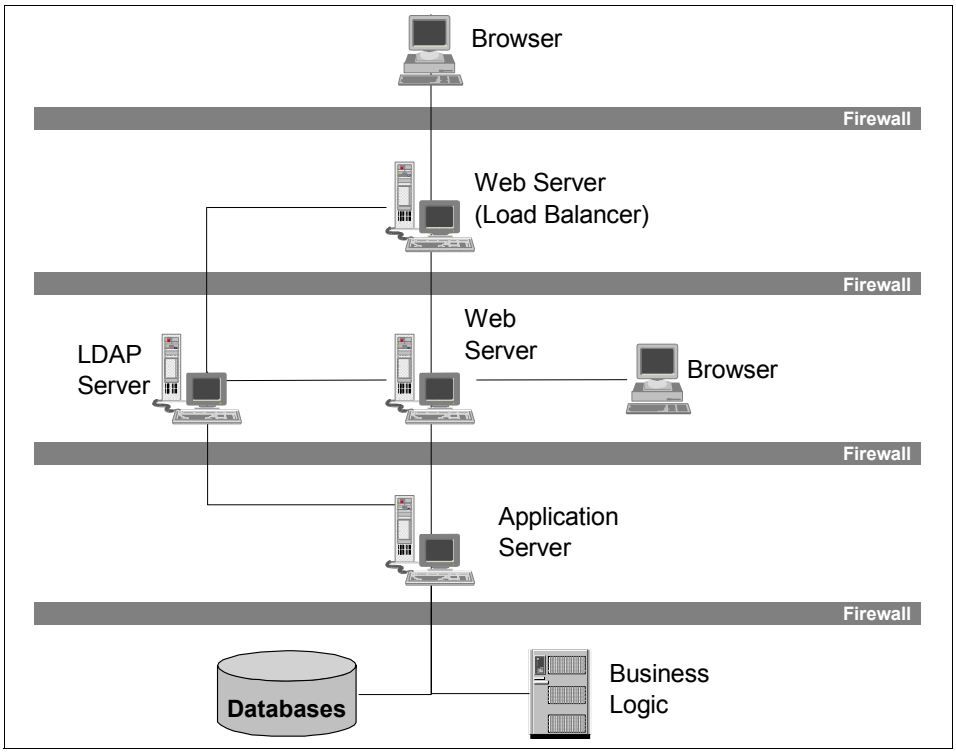


Figure 1-8 Logical view of an e-business solution

For more information on e-business architectures, please refer to the redbook *Patterns for e-business: User to Business Patterns for Topology 1 and 2 Using WebSphere Advanced Edition*, SG24-5864, which can be downloaded from http://www.redbooks.ibm.com®.

Tivoli and IBM provide some of the most widely used products to implement the e-business infrastructure. These are:

IBM HTTP Server Communication and transaction control

Tivoli Access Manager
Identification, authentication, and authorization

IBM WebSphere Application Server
Web application hosting, responsible for the transformation services

IBM WebSphere Edge Server
Web application firewalling, load balancing, Web hosting; responsible for the transformation services

1.3.4 Managing e-business applications using Tivoli

Even though the e-business patterns help in designing e-business applications by breaking them down into functional units that may be implemented in different tiers of the architecture using different hard- and software technologies, the patterns provide only some assistance in managing these applications. Fortunately, this gap is filled by solutions from Tivoli Systems.

When designing the systems management infrastructure that is needed to manage the e-business applications, it must be kept in mind that the determining factor for the application architecture is the nature of the application itself. This determines the application infrastructure and the technologies used. However, it does not do any harm if the solution architect consults with systems management specialists while designing the application.

The systems management solution has to play more or less by the rules set up by the application. Ideally, it will manage the various application resources without any impact on the e-business application, while observing company policies on networking use, security, and so on.

Management of e-business applications is therefore best achieved by establishing yet another networking tier, parallel to the application tier, in which all systems management components can be hosted without influencing the applications. Naturally, since the management applications have to communicate with the resources that must be managed, the two meet on the network and on the machines hosting the various e-business application resources.

Using the Tivoli product set, it is recommended that you establish all the central components in the management tier and have a few proxies and agents present in the DMZ and application tiers, as shown in Figure 1-9 on page 27.

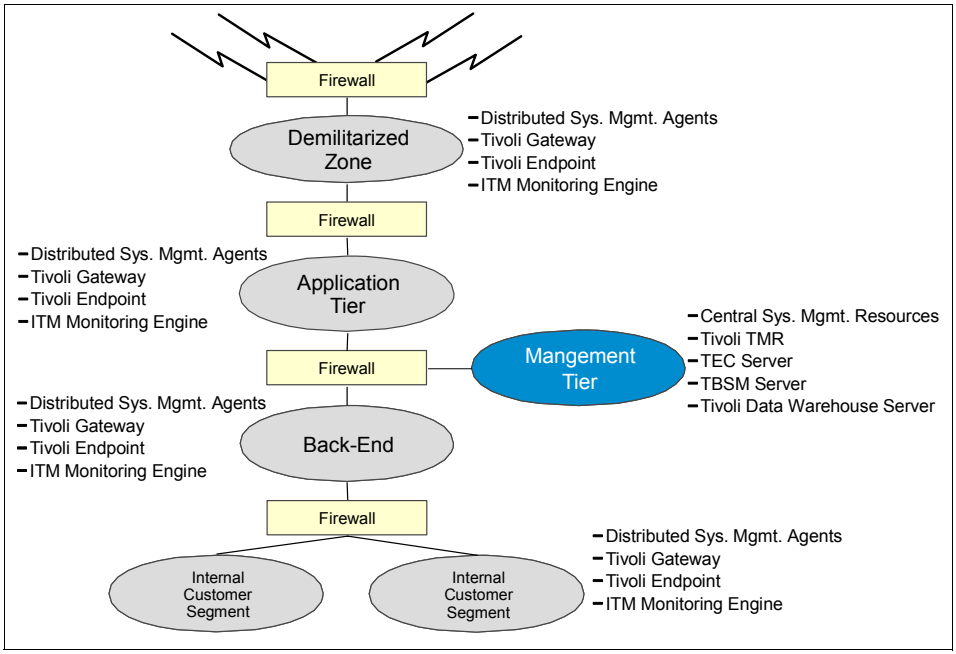


Figure 1-9 Typical Tivoli-managed e-business application infrastructure

Implementing the management infrastructure in this fashion, there is minimal interference between the application and the management systems, and the access to and from the various network segments is manageable, as the communication flows between a limited number of nodes using well-known communication ports.

IBM Tivoli management products have been developed with the total environment in mind. The IBM Tivoli Monitoring product provides the basis for proactive monitoring, analysis, and automated problem resolution.

As we will see, IBM Tivoli Monitoring for Transaction Performance provides an enterprise management solution for both the Web and enterprise transaction environments. This product provide solutions that are integrated with other Tivoli management products and contribute a key piece to the goal of a consistent, end-to-end management solution for the enterprise.

By using product offerings such as IBM Tivoli Monitoring for Transaction Performance in conjunction with the underlying Tivoli technologies, a comprehensive and fully integrated management solution can be deployed rapidly and provide a very attractive return on investment.

1.4 Tivoli product structure

Let us take a look at how Tivoli solutions provide comprehensive systems management for the e-business enterprise and how the IBM Tivoli Monitoring for Transaction Performance product fits into the overall architecture.

In the hectic *on demand* environments e-businesses find themselves in today, responsiveness, focus, resilience, and variability/flexibility are key to conducting business successfully. Most business processes rely heavily on IT systems, so it is fair to say that the IT systems have to possess the same set of attributes in order to be able to keep up with the speed of business. To provide an open framework for the on demand IT infrastructure, IBM has published the On Demand Blueprint, which defines an On Demand Operating Environment with three major properties (Figure 1-10):

Integration Efficient and flexible combination of resources (people, processes, and information) to optimize resources across and beyond the enterprise.

Automation The capability to dynamically deploy, monitor, manage, and protect an IT infrastructure to meet business needs with little or no human intervention.

Virtualization
Present computer resources in ways that allows users and applications to easily get value out of them, rather than presenting them in ways dictated by the implementation, geographical location, or physical packaging.

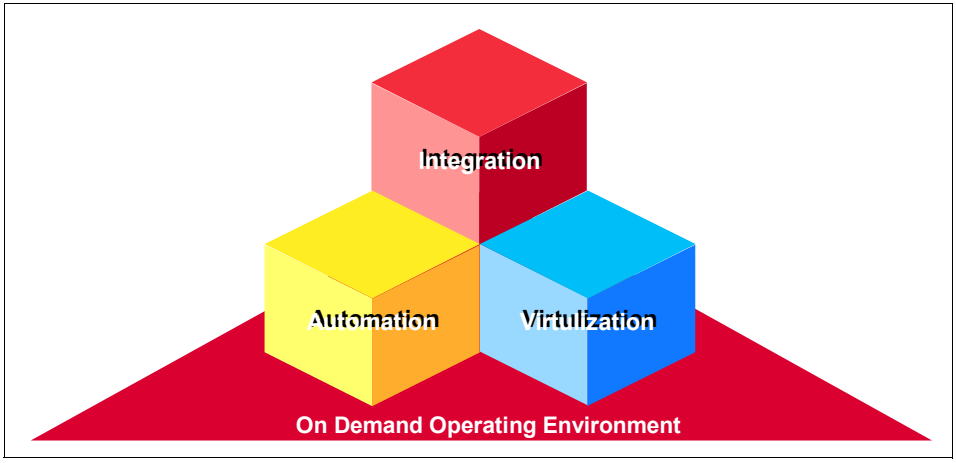


Figure 1-10 The On Demand Operating Environment

The key motivators for taking steps to align the IT infrastructure with the ideas of the On Demand Operating Environment are:

Align the IT processes with business priorities
Allow your business to dictate how IT operates, and eliminate constraints that prohibits the effectiveness of your business.

Enable business flexibility and responsiveness
Speed is the one of the critical determinants of competitive success. IT processes that are too slow to keep up with the business climate cripples corporate goals and objectives. Rapid response and nimbleness mean that IT becomes an enabler of business advantage versus a hindrance.

Reduce cost
By increasing the automation in your environment, immediate benefits can be realized from lower administrative costs and less reliance on human operators.

Improved asset utilization
Use resources more intelligently. Deploy resources on an as-needed, just-in-time basis, versus a costly and inefficient "just-in-case" basis.

Address new business opportunities
Automation removes lack of speed and human error from the cost equation. New opportunities to serve customers or offer better services will not be hampered by the inability to mobilize resources in time.

In the On Demand Operating Environment, IBM Tivoli Monitoring for Transaction Performance plays an important role in the automation area. By providing functions to determine how well the users of the business transactions (the J2EE based ones in particular) are served, IBM Tivoli Monitoring for Transaction Performance supports the process of provisioning adequate capacity to meet Service Level Objectives, and helps automate problem determination and resolution.

For more information on the IBM On Demand Operation Environment, please refer to the Redpaper *e-business On Demand Operating Environment*, REDP3673.

As part of the On Demand Blueprint, IBM provides specific Blueprints for each of the three major properties. The IBM Automation Blueprint depicted in Figure 1-11 on page 30 defines the various components needed to provide automation services for the On Demand Operation Environment.

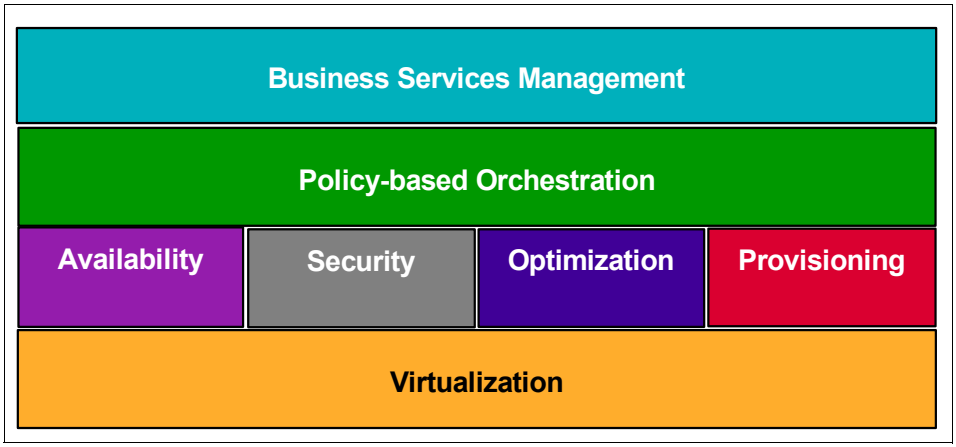


Figure 1-11 IBM Automation Blueprint

The IBM Automation Blueprint defines groups of common services and infrastructure that provide consistency across management applications, as well as enabling integration.

Within the Tivoli product family, there are specific solutions that target the same five primary disciplines of systems management:

- Availability
- Security
- Optimization
- Provisioning
- Policy-based Orchestration

Products within each of these areas have been made available over the years and, as they are continually enhanced, have become accepted solutions in enterprises around the world. With these core capabilities in place, IBM has been able to focus on building applications that take advantage of these solution-silos to provide true business systems management solutions.

A typical business application depends not only on hardware and networking, but also on software ranging from the operating system to middleware such as databases, Web servers, and messaging systems, to the applications themselves. A suite of solutions such as the “IBM Tivoli Monitoring for...” products, enables an IT department to provide consistent availability management of the entire business system from a central site and using an integrated set of tools. By utilizing an end-to-end set of solutions built on a common foundation, enterprises can manage the ever-increasing complexity of their IT infrastructure with reduced staff and increased efficiency.

Within the availability group in Figure 1-11 on page 30, two specific functional areas are used to organize and coordinate the functions provided by Tivoli products. These areas are shown in Figure 1-12.

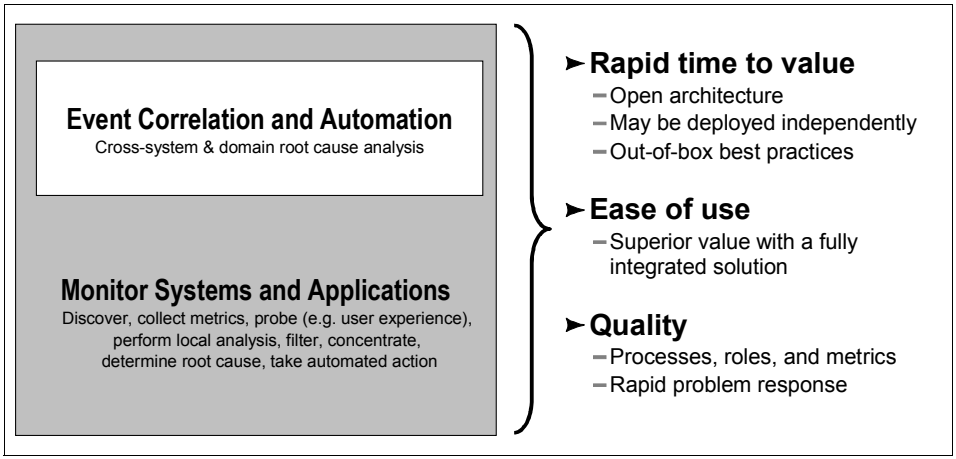

Event Correlation and Automation
Cross-system & domain root cause analysis
Monitor Systems and Applications
Discover, collect metrics, probe (e.g. user experience), perform local analysis, filter, concentrate, determine root cause, take automated action
Rapid time to value
Open architecture
May be deployed independently
Out-of-box best practices
Ease of use
Superior value with a fully integrated solution
Quality
Processes, roles, and metrics
Rapid problem response

Figure 1-12 Tivoli's availability product structure

The lowest level consists of the monitoring products and technologies, such as IBM Tivoli Monitoring and its resource models. At this layer, Tivoli applications monitor the hardware and software and provide automated corrective actions whenever possible.

At the next level is event correlation and automation. As problems occur that cannot be resolved at the monitoring level, event notifications are generated and sent to a correlation engine, such as Tivoli Enterprise Console®. The correlation engine at this point can analyze problem notifications (events) coming from multiple components and either automate corrective actions or provide the necessary information to operators who can initiate corrective actions.

Both tiers provide input to the Business Information Services category of the Blueprint. From a business point-of-view, it is important to know that a component or related set of components has failed as reported by the monitors in the first layer. Likewise, in the second layer, it is valuable to understand how a single failure may cause problems in related components. For example, a router being down could cause database clients to generate errors if they cannot access the database server. The integration to Business Information Services is a very important aspect, as it provides an insight into how a component failure may be affecting the business as a whole. When the router failure mentioned above occurs, it is important to understand exactly what line of business applications will be affected and how to reduce the impact of that failure on the business.

1.5 Managing e-business applications

As we have seen, managing e-business applications requires that basic services such as communications, messaging, database, and application hosting are functional and well-behaved. This should be ensured by careful management of the infrastructural components using Tivoli tools to facilitate monitoring, event forwarding, automation, console services, and business impact visualization.

However, ensuring the availability and performance of the application infrastructure is not always enough. Web-based applications are implemented in order to attract business from customers and business partners who we may or may not know. Depending on the nature of the data provided by the application, company policies for security and access control, as well as access to and use of specific applications, may be restricted to users whose identity can be authenticated. In other instances (for example, online news services), there are user authentication requirements for access to the application.

In either case, the goal of the application is to provide useful information to the user and, of course, attract the user to return later. The service provided to the user, in terms of functionality, ease of use, and responsiveness of the application, is critical to the user's perception of the application's *usefulness*. If the user finds the application useful, there is a fair chance that the user will return to conduct more business with the application owner.

The usefulness of an application is a very subjective measure, but it seems fair to assume that an individual's perception of an application's usefulness involves, at the very least:

- Relevance to current needs
- Easy-to-understand organization and navigation
- Logical flow and guidance
- The integrity of the information (is it trustworthy?)
- Responsiveness of the application

Naturally, the application owner can influence all of these parameters (the application design can be modified, the data can be validated, and so on) but network latency and the capabilities of the user's system are critical factors that may affect the time it takes for the user to receive a response from the application. To avoid this becoming an issue that scares users away from the application, the application provider can:

- Set the user's expectations by providing sufficient information up front.
- Make sure that the back-end transaction performance is as fast as possible.

Neither of these will guarantee that users will return to the application, but monitoring and measuring the total response time and breaking it down into the

various components shown in Figure 1-1 on page 4 will give the application owner an indication of where the bottlenecks might be.

To provide consistently good response times from the back-end systems, the application provider may also establish a monitoring system that generates reference transactions on a scheduled basis. This will give early indications about upcoming problems or adjust the responsiveness of the applications.

The need for real-time monitoring and gathering of reference (and historical) data, among others, are addressed by IBM Tivoli Monitoring for Transaction Performance. By providing the tools necessary for understanding the relationships between the various components that make up the total response time of an application, including breakdown of the back-end service times into service times for each subtransaction, IBM Tivoli Monitoring for Transaction Performance is the tool of choice for monitoring and measuring transaction performance.

1.5.1 IBM Tivoli Monitoring for Transaction Performance functions

IBM Tivoli Monitoring for Transaction Performance provides functions to monitor e-business transaction performance in a variety of situations. Focusing on e-business transactions, it should come as no surprise that the product provides functions for transaction performance measurement for various Web-based transaction types originating from external systems (systems situated somewhere on the Internet and not managed by the organization) that provide the e-business transactions or applications that are the target of the performance measurement. These transactions are referred to in the following pages as *Web transactions*, and they are implemented by the Web Transaction Performance module of IBM Tivoli Monitoring for Transaction Performance.

In addition, a set of functions specifically designed to monitor the performance metrics of transactions invoked from within the corporate network (known as *enterprise transactions*) are provided by the product's Enterprise Transaction Performance module. The main function of Enterprise Transaction Performance is to monitor transaction performance of applications that have transaction performance probes (ARM calls) included. In addition, Enterprise Transaction Performance provides functions to monitor online transactions with mainframe sessions (3270) and SAP systems, non-Web based response times for transactions with mail and database servers, and Web-based transactions with HTTP servers, as shown in Figure 1-13 on page 34.

It should be noted that the tools for Web and enterprise transaction performance monitoring complement one another, and that there are no restrictions, if the networking and management infrastructure is in place, for using Enterprise monitors in the Web space or vice versa.

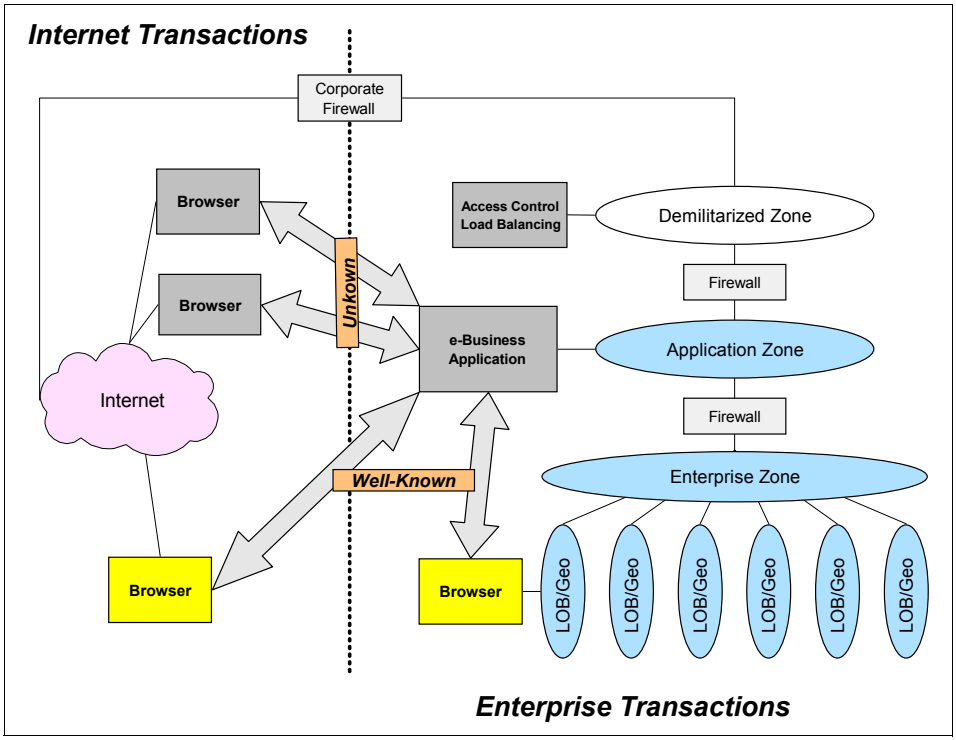


Figure 1-13 e-business transactions

Web transaction monitoring

In general, the nature of Web transaction performance measurement is random and generic. There is no way of planning the execution of transactions or the origin of the transaction initiation unless other measures have been taken in order to do so. When the data from the transaction performance measurements are being aggregated, they provide information about the *average* transaction invocation, without affinity to location, geography, workstation hardware, browser version, or other parameters that may affect the experience of the end user. All of these parameters are out of the application provider's control. Naturally, both the data gathering and reporting may be set up to only handle transaction performance measurements from machines that have specific network addresses, for example, thus limiting the scope of the monitoring to well-known machines. However, the transactions executed, and the sequence is still random and unplanned.

The monitoring infrastructure used to capture performance metrics of the average transaction may also be used to measure transaction performance for specific, pre-planned transactions initiated from well-known systems accessing

the e-business applications through the Internet or intranet. To facilitate this kind of *controlled* measurements, certain programs must be installed on the systems initiating the transactions, and they will have to be controlled by the organization that wants the measurements. From a transaction monitoring point of view there are no differences between monitoring *average* or *controlled* transactions; the same data may be gathered to the same level of granularity. The big difference is that the monitoring organization knows that the transaction is being executed, as well as the specifics of the initiating systems.

The main functions provided by IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance are:

- For both unknown and well-known systems:
  - Real-time transaction performance monitoring
  - Transaction breakdown
  - Automatic problem identification and baselining
- For well-known systems with specific programs installed:
  - Transaction simulation based on recording and playback
  - Web transaction availability monitoring

Enterprise transaction monitoring

If the application provider wants to gather transaction performance characteristics from workstations situated within the enterprise network or machines that are part of the managed domain, but initiates transactions through the Internet, a different set of tools is available. These are provided by the Enterprise Transaction Performance module of the IBM Tivoli Monitoring for Transaction Performance product.

The functions provided by Enterprise Transaction Performance are integrated with the Tivoli Management Environment and rely on common services provided by the integration. Therefore, the systems from which transaction performance data is being gathered must be part of the Tivoli Management Environment, and at a minimum have a Tivoli endpoint installed. This will, however, enable centralized management of the systems for additional functions besides the gathering of transaction performance data.

In addition to monitoring transactions initiated through a browser, just like the ones we earlier called Web transactions, Enterprise Transaction Performance provides specialized programs, end-to-end probes, which enable monitoring of the time needed to load a URL and specific transactions related to certain mail and groupware applications. The Enterprise module also provides unique recording and playback functions for transaction simulation of 3270 and SAP

applications, and a generic recording/playback solution to be used only on Windows®-based systems.

2

IBM Tivoli Monitoring for Transaction Performance in brief

This chapter provides a high level overview of the functionality incorporated in IBM Tivoli Monitoring for Transaction Performance Version 5.2. We also introduce some of the reporting capabilities provided by TMTP.

© Copyright IBM Corp. 2003. All rights reserved.

2.1 Typical e-business transactions are complex

Figure 2-1 depicts a typical e-business application. Typically, it will involve multiple firewalls and an application that will have many components distributed across many different servers.

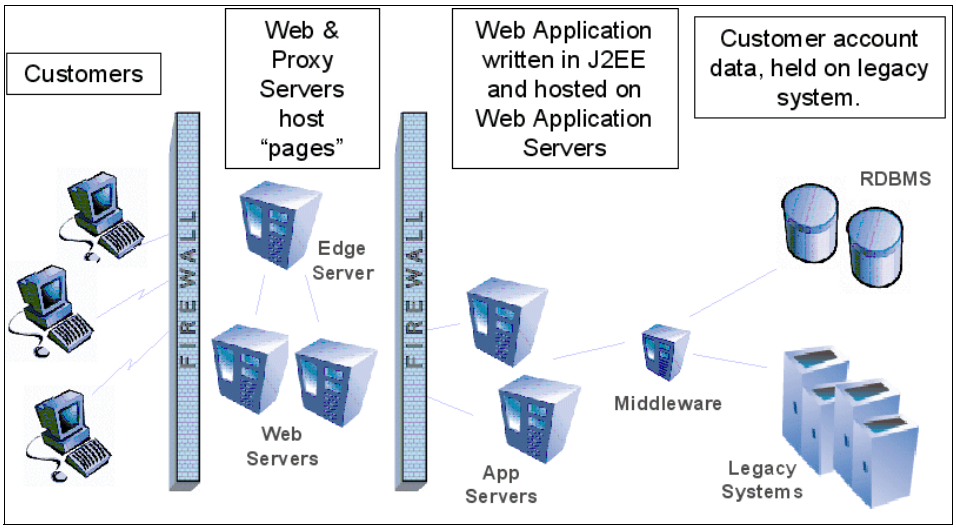


Figure 2-1 Typical e-business transactions are complex

As you can tell from Figure 2-1, there are also multiple machines doing the same piece of work (as is indicated by the duplication of the Web servers, application servers, and databases). This level of duplication is needed to ensure high availability and to handle a large number of concurrent users. The architecture that you see here is different in several ways from the past. In the past, all of these components were often on a single infrastructure (the mainframe). This all changed with the evolution of client server, and is now changing again with the trend towards Web Services.

2.1.1 The pain of e-business transactions

Generally, when monitoring an environment such as that described above, the response to a customer complaint about poor performance can be described as follows:

Step 1 Typically, a call comes in to the help desk indicating that the response time for your e-business application is unacceptable.

This is the first place where you need a transaction performance

product (to find out if there is a problem, hopefully before the customer calls you to identify a problem).

Important: At step 1, if the customer has IBM Tivoli Monitoring, then they would see far few problems even show up, because they are being automatically cured by resource models. If the customer has TBSM, and it is a resource problem, then there is a good chance that the team is already working on solving the problem if it is in a critical place.

Step 2 The next step usually involves the operations center. The Network Operations Center (NOC) gets the message and starts by looking at the network to see if they can detect any problems at this level.

Operations team in the NOC calls the SysAdmins (or Senior Technical Support Staff, that is, the more senior staff that are responsible for applications in production).

Step 3 Then a lot of people are paged! The number of pagers that go off is often dependent on the severity of the SLA or the customer involved. If it is a big problem, a “tiger team” will be assembled. This typically large group of people are assembled to try and resolve the problem.

Step 4 The SysAdmins check to see if anything has changed in the past day to understand what the cause may be. If possible, they roll back to a previous version of the application to see if that fixes the problem.

The SysAdmins then typically have a check list of things they do or tools they use to troubleshoot the problem. Some of the tasks they may perform are:

- Look at any monitoring tools for hardware, OS, and applications.
- Look at the packet data: number of collisions, loss between connections, and so on.
- Crawl through the log files from the application, middleware, and so on.
- The DBAs will check databases from the command line to see what response time looks like from there.
- Call other parties that may be related (host based applications, application developers that maintain the application, and so on).

Step 5 Finger pointing. Unfortunately, it is still very difficult to solve the problem. These tiger teams often generate a lot of finger pointing and blaming. This is unpleasant and itself leads to longer problem resolution response times.

All of this is very painful and can be very expensive.

TMTP 5.2 solves this problem by pinpointing the exact cause of a transaction performance problem with your e-business application quickly and easily, and then facilitating resolution of that problem.

2.2 Introducing TMTP 5.2

IBM Tivoli Monitoring for Transaction Performance Web Transaction Performance (TMTP WTP) is a centrally managed suite of software components that monitor the availability and performance of Web-based services and Microsoft® Windows applications. IBM Tivoli Monitoring for Transaction Performance captures detailed performance data for all of your e-business transactions. You can use this software to perform the following e-business management tasks:

- Monitor every step of an actual customer transaction as it passes through the complex array of hosts, systems, and applications in your environment: Web and proxy servers, Web application servers, middleware, database management systems, and legacy back-office systems and applications.
- Simulate customer transactions, collecting “what if?” performance data that helps you assess the health of your e-business components and configurations.
- Consult comprehensive real-time reports that display recently collected data in a variety of formats and from a variety of perspectives.
- Integrate with the Tivoli Enterprise Date Warehouse, where you can store collected data for use in historical analysis and long-term planning.
- Receive prompt, automated notification of performance problems. With IBM Tivoli Monitoring for Transaction Performance, you can effectively measure how users experience your Web site and applications under different conditions and at different times. Most important, you can quickly isolate the source of performance problems as they occur, so that you can correct those problems before they produce expensive outages and lost revenue.

2.2.1 TMTP 5.2 components

IBM Tivoli Monitoring for Transaction Performance provides the following major components that you can use to investigate and monitor transactions in your environment.

Discovery component

The discovery component enables you to identify incoming Web transactions that need to be monitored.

Two listening components

Listening components collect performance data for actual user transactions that are executed against the Web servers and Web application servers in your environment. For example, you can use a listening component to gauge the time it takes for customers to access an online product catalog and order a specific item. Listening components, also called listeners, are the *Quality of Service* and *J2EE monitoring* components.

Two playback components

Playback components robotically execute, or play back, transactions that you record in order to simulate actual user activity. For example, you can record and play back an online ordering transaction to assess the relative performance of different Web servers, or to identify potential bottlenecks before launching a new interactive application. Playback components are *Synthetic Transaction Investigator* and *Rational® Robot/Generic Windows*.

Discovery, listening, and playback operations are run according to instructions set forth in policies that you create. A policy defines the area of your Web site to investigate or the transactions to monitor, indicates the types of information to collect, specifies a schedule, and provides a range of other parameters that determine how and when the policy is run.

The following subsections describe the discovery, listening, and playback components.

The discovery component

When you use the discovery process, you create a discovery policy in which you define an area of your Web environment that you want to investigate. The discovery policy then samples transaction activity and produces a list of all URI requests, with average performance times, that have occurred during a discovery period. You can consult the list of discovered URIs to identify transactions to monitor with listening policies.

A discovery policy is associated with one of the two listening components. A Quality of Service discovery policy discovers transactions that run through the Web servers in your environment. A J2EE discovery policy discovers transactions that run on J2EE application servers. Figure 2-2 on page 42 shows an example of a discovered application topology.

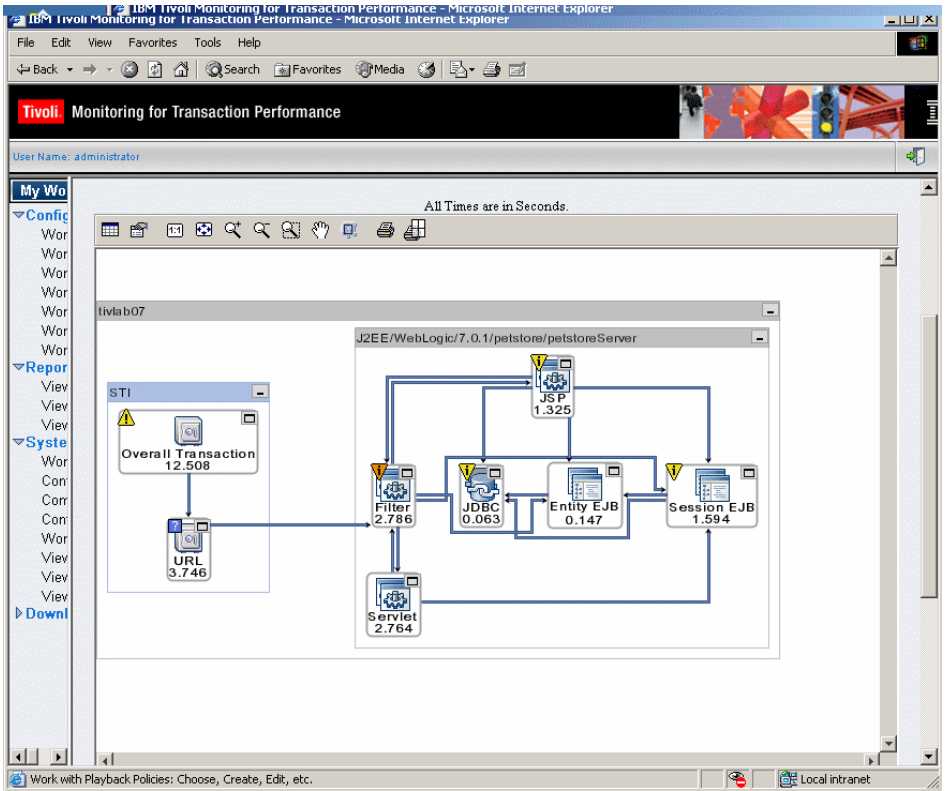


Figure 2-2 Application topology discovered by TMTP

Listening: The Quality of Service component

The Quality of Service component samples incoming HTTP transactions against a Web server and measures various time intervals involved in completing each transaction. An HTTP transaction consists of a single HTTP request and response.

A sample of transactions might consist of every tenth transaction from a specific collection of users over a peak time period. The Quality of Service component can measure the following time intervals for each transaction:

- Back-end service time. This is the time it takes a Web server to receive the request, process it, and respond to it.
- Page render time. This is the time it takes to process and display a Web page on a browser.
- Round-trip time (also called user experience time). This is the time it takes to complete the entire page request, from the moment the user initiates the

request (by clicking on a link, for example) until the request is fulfilled. Round-trip time includes back-end service time, page render time, and network and data transfer time.

Listening: The J2EE monitoring component

The J2EE monitoring component collects performance data for transactions that run on a J2EE (Java 2 Platform Enterprise Edition) application server. Six J2EE subtransaction types can be monitored: servlets, session beans, entity beans, JMS, JDBC, and RMI. The J2EE monitoring component supports the following two application servers:

- IBM WebSphere Application Server 4.0.3 and up
- BEA WebLogic 7.0.1

You can dynamically install and remove ARM instrumentation for either type of application server. You can also enable and disable the instrumentation.

Playback: Synthetic Transaction Investigator

The Synthetic Transaction Investigator (STI) component measures how users might experience a Web site in the course of performing a specific transaction, such as searching for information, enrolling in a class, or viewing an account. Using STI involves the following two activities:

- Recording a transaction. You use STI Recorder to record your actions as you perform the sequence of steps that make up the transaction. For example, you might perform the following steps to view an account: log on, click to display the main menu, click to view an account summary and log off. The mechanism for recording is to save all HTTP request information in an XML document.
- Playing back the transaction. STI plays back the recorded transaction according to parameters you specify. You can schedule a playback to repeat at different times and from different locations in order to evaluate performance and availability under varying conditions. During playback, STI can measure response times, check for missing or damaged links, and scan for specified content.

Playback: Rational Robot/Generic Windows

Together, Rational Robot and Generic Windows enable you to gauge how users might experience a Microsoft Windows application that is used in your environment. Like STI, Rational Robot and Generic Windows involve record and playback activities:

- Recording a transaction. You use Rational Robot to record the application actions that you want to investigate. For example, you might record the actions involved in accessing a proprietary document sharing application

deployed on an application server. The steps might include logging on and obtaining the main page display.

- Playing back the transaction. The Generic Windows component plays back the recorded transaction and measures response times.

2.3 Reporting and troubleshooting with TMTP WTP

One of the strengths of this release of TMTP is its reporting capabilities. The following subsections introduce you to the various visual components and reports that can be gathered from TMTP and the way in which these could be used.

Troubleshooting transactions with the Topology view

Your organization has installed TMTP V5.2 and it has been configured to send e-mail to the TMTP Administrator as well as sending an event to the Tivoli Enterprise Console upon a transaction performance violation. Using the following steps, the TMTP administrator identifies and analyzes the transaction performance violation and ultimately identifies the root cause.

After receiving the notification from TMTP, the Administrator would log onto TMTP and access the “Big Board” view, shown in Figure 2-3.

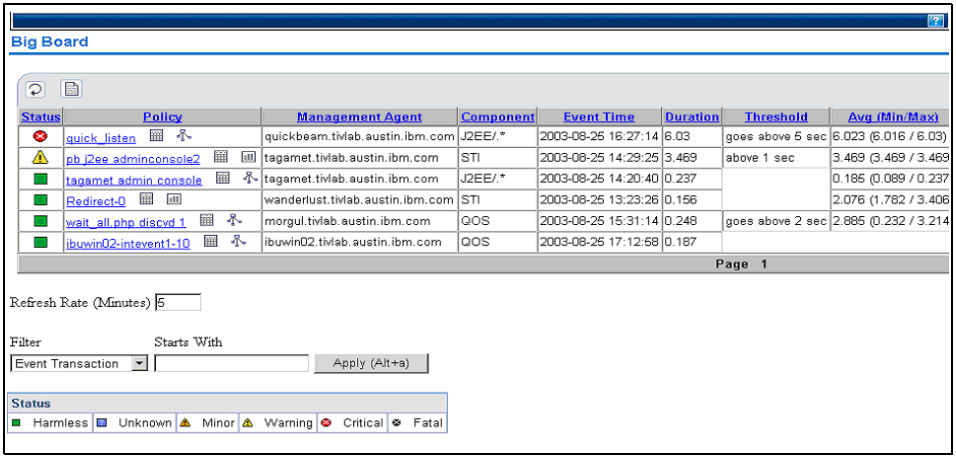


Figure 2-3 Big Board View

From the Big Board View, the administrator can see that the J2EE policy called “quick_listen” had a violation at 16:27. The user can also tell the policy had a threshold of “goes above 5 seconds”, which was violated, as the value was 6.03 seconds.

The administrator can now click on the topology icon for that policy and load the most recent topology that TMTP has data for (see Figure 2-4).

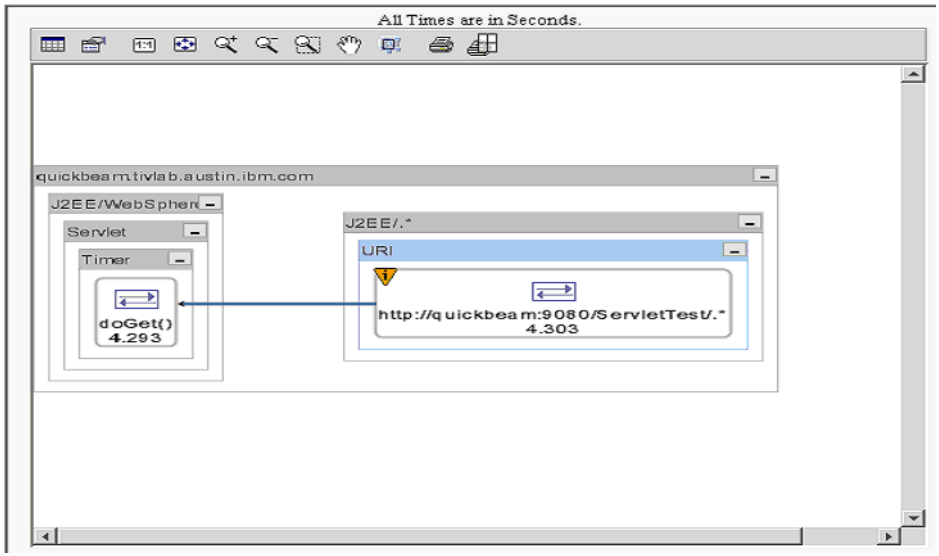


Figure 2-4 Topology view indicating problem

Since, by default, topologies are filtered to exclude any nodes that are slower than one second (this is configurable), the default view is to show the latest aggregated data for slow nodes. In Figure 2-4, you can see that there were only two slow performing nodes.

All nodes in the topology have a numeric value on them. If the node is a container for other nodes (for example, a Servlet node may contain four different Servlets) the time expressed on the node is the maximum time of what is contained within the node. This makes it easy to track down where the slow node resides. Once you have drilled down to the bottom level, the time on the base node indicates the actual time for that node (average for aggregate data, and specific timings for instance data). In Figure 2-4, the root node (J2EE/.*) has an icon that indicates that it has had performance violations for that hour.

The administrator can now select the node that is in violation and click on the **Inspector** icon. The Inspector view (Figure 2-5 on page 46) reveals that the threshold setting of “goes above 5 seconds” was violated nine times out of 11 for the hour and that the minimum time was 0.075 and the maximum time was 6.03. The administrator can conclude from these numbers that this nodes performance was fairly erratic.

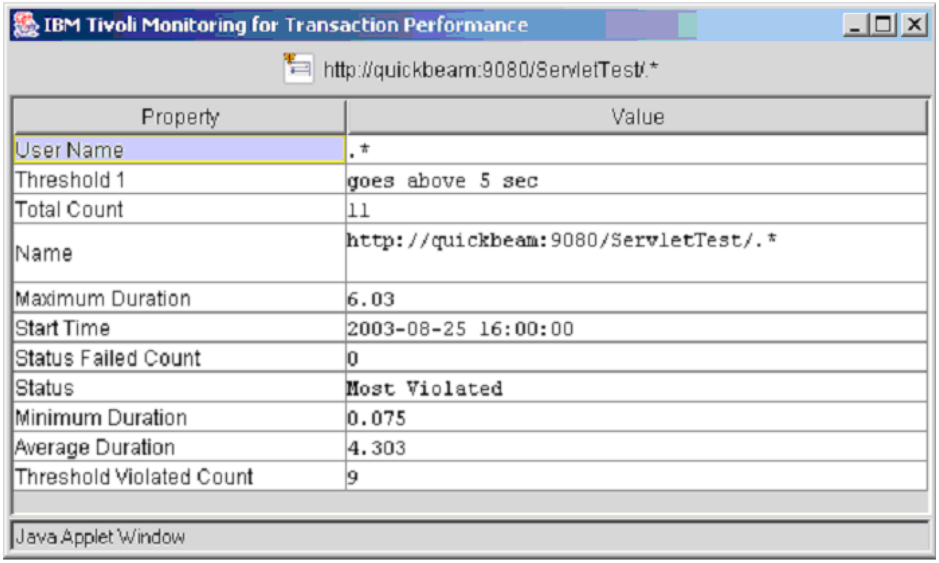


Figure 2-5 Inspector view

By examining the instance drop-down list (Figure 2-6), the administrator can see all of the instances captured for the hour.

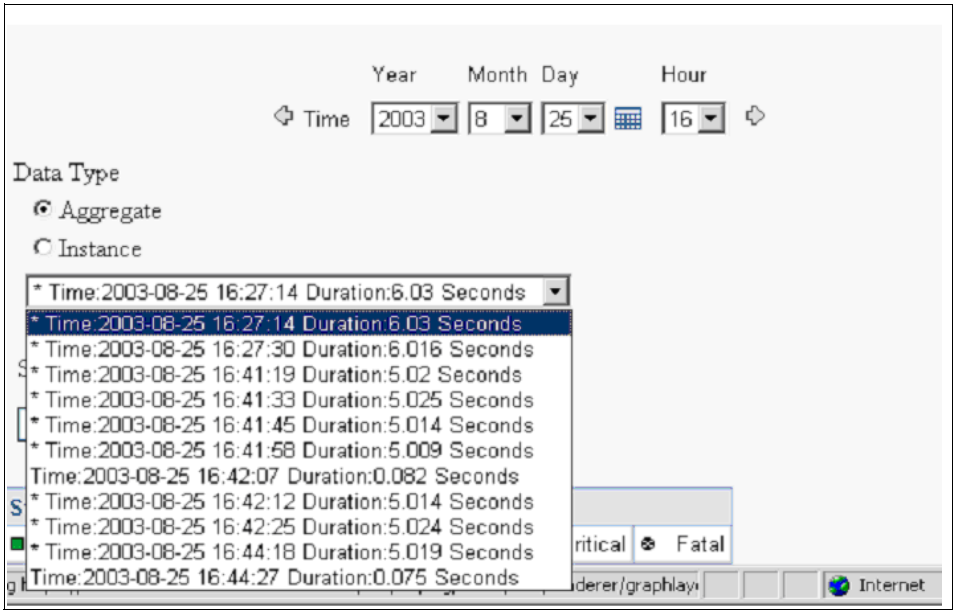


Figure 2-6 Instance drop down

Figure 2-6 on page 46 shows nine instances with asterisks indicating that they violated thresholds and two others with low performance figures indicating they did not violate. The administrator can now select the first instance that violated (they are in order of occurrence) and click the **Apply** button to obtain an instance topology (Figure 2-7).

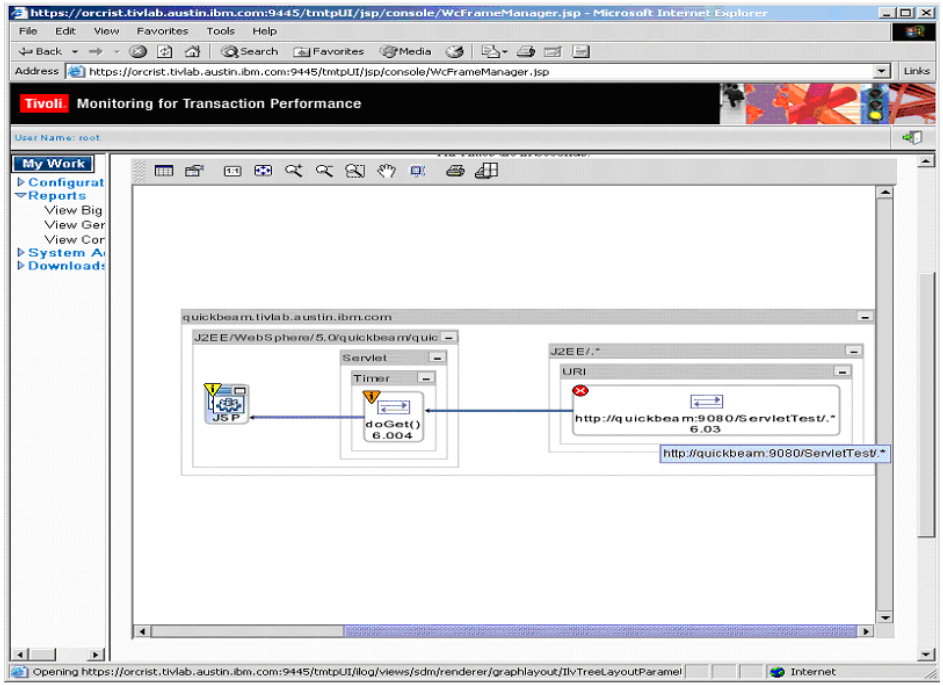


Figure 2-7 Instance topology

Again, this topology has the one second filtering turned on, so any extraneous nodes are filtered out. Here the administrator can see that, as suspected, the Timer.goGet() method is taking up a majority of the time, ruling out a problem with the root transaction.

The Timer.goGet() method has an upside down orange triangle indicating it has been deemed the most violated instance. This calculation is determined by comparing the instances duration (6.004 seconds in this case) to the average for the hour (4.303 seconds, as we saw above) while taking into account the number of times the method was called by that method. Doing this provides an estimate of the amount of time spent in a node that was above its average. This calculation provides an indication of abnormal behavior because it is slower than normal. Other slow performing nodes will be marked with a yellow upside down triangle, indicating a problem against the average for the hour (by default, 5% of the methods will have a marking).

Selecting the Timer.doGet() node and examining the inspector would show any metrics captured for the Servlet. In this example, the Servlet tracing is minimal, and the following figure is what would be displayed by the inspector (Figure 2-8). If greater tracing were specified, the context metrics could provide information on SQL statements, login information, and so on (some of the later chapters will demonstrate this), depending on the type of node selected and the level of tracing configured in the listening policy.

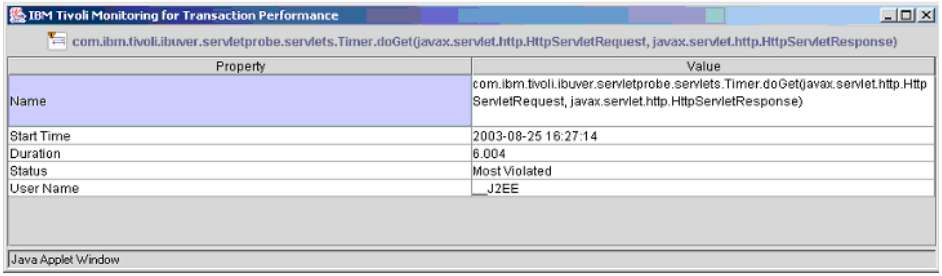


Figure 2-8 Inspector viewing metrics

Using these steps, the administrator has very quickly determined that the cause of the poor performance is a particular servlet, and the root cause is a specific method (Timer.doGet()) of that servlet. Narrowing the problem down this quickly to a component of an application would previously have taken a lot of time and effort, if it was ever discovered at all. Often, it is all just a little too hard to find the problem, and the temptation is to buy more hardware. This administrator has just saved his organization the expense of purchasing additional hardware because of a poorly performing servlet method.

Other reports provided with TMTP

Some of the other reports available from within TMTP are shown in this section.

Overall Transactions Over Time

This report (Figure 2-9 on page 49) can be used to investigate the performance of a monitored transaction over a specified period of time.

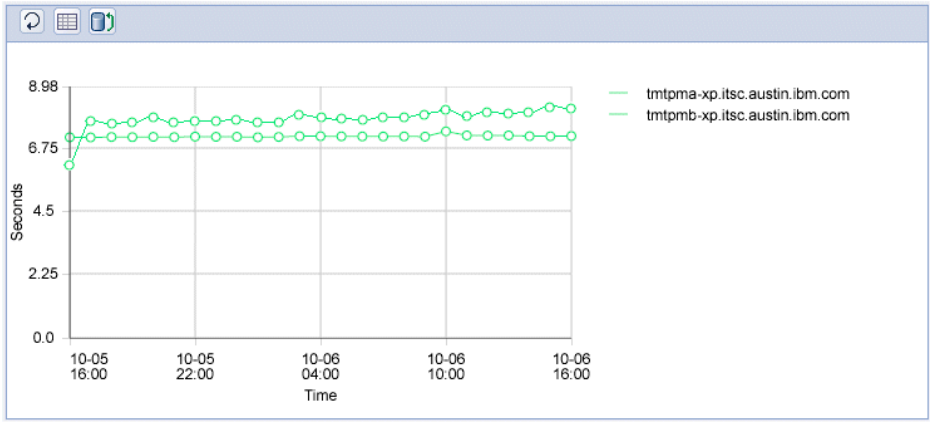


Figure 2-9 Overall Transactions Over Time

Transactions with Subtransactions

This report (Figure 2-10 on page 50) can be used to investigate the performance of a monitored transaction and up to five of its subtransactions over a specified period of time. A line with data points represents the aggregate response times collected for a specific transaction (URI or URI pattern) that is monitored by a specific monitoring policy running on a specific Management Agent. Colored areas below the line represent response times for up to five subtransactions of the monitored transaction. When a transaction is considered together with its subtransactions, as it is in this graph, it is often referred to as a parent transaction. Similarly, the subtransactions are referred to as children of the parent transaction.

By default, when you open the Transactions With Subtransactions graph, the display shows the parent transaction with the highest recent aggregate response times. The default graph also shows the five subtransaction children with the highest response times. You can specify a different transaction for the display, and you can also specify any subtransactions of the specified transaction. In addition, you can manipulate graph contents in a variety of other ways to see precisely the data that you want to view.

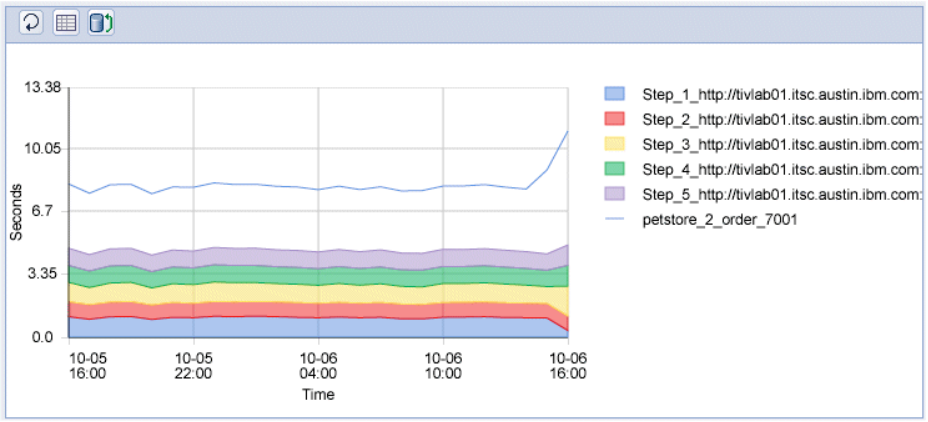


Figure 2-10 Transactions with Subtransactions

Page Analyzer Viewer

The Page Analyzer Viewer Report window (Figure 2-11) allows you to view the performance of Web screens that are visited during a synthetic transaction. The Page Analyzer Viewer Report window gives details about the timing, size, identity, and source of each item that makes up a page. You can use this information to evaluate Web page design regarding efficiency, organization, and delivery.

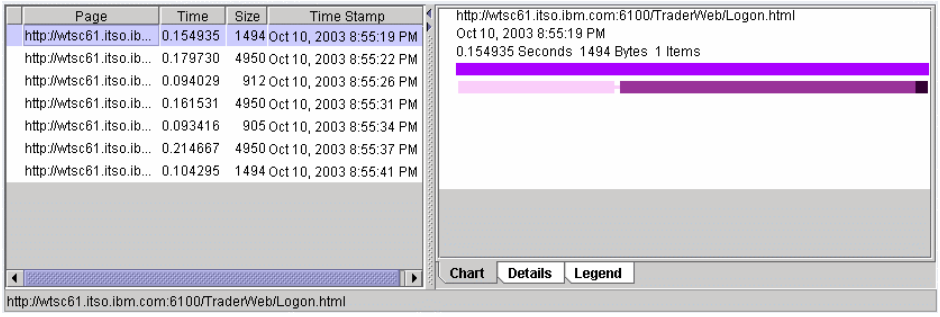

Page
Time
Size
Time Stamp
http://wtsc61.itso.ib... 0.154935 1494 Oct 10, 2003 8:55:19 PM
http://wtsc61.itso.ib... 0.179730 4950 Oct 10, 2003 8:55:22 PM
http://wtsc61.itso.ib... 0.094029 912 Oct 10, 2003 8:55:26 PM
http://wtsc61.itso.ib... 0.161531 4950 Oct 10, 2003 8:55:31 PM
http://wtsc61.itso.ib... 0.093416 905 Oct 10, 2003 8:55:34 PM
http://wtsc61.itso.ib... 0.214667 4950 Oct 10, 2003 8:55:37 PM
http://wtsc61.itso.ib... 0.104295 1494 Oct 10, 2003 8:55:41 PM
http://wtsc61.itso.ibm.com:6100/TraderWeb/Logon.html
Oct 10, 2003 8:55:19 PM
0.154935 Seconds 1494 Bytes 1 Items
Chart
Details
Legend
http://wtsc61.itso.ibm.com:6100/TraderWeb/Logon.html

Figure 2-11 Page Analyzer Viewer

A more detailed introduction to the reporting capabilities of TMTP is included in Chapter 7, “Real-time reporting” on page 211. Historical reporting using the Tivoli Data Warehouse is covered in Chapter 10, “Historical reporting” on page 375. Additionally, several of the chapters include scenarios that show how to use the reporting capabilities of the TMTP product in order to identify e-business transaction problems. This is important, as the dynamic nature and

drill down capabilities of reports (such as the Topology overview) are very powerful problem solving and troubleshooting tools.

2.4 Integration points

Existing IBM Tivoli Customers are aware of the value that can be obtained by integrating IBM Tivoli products into a complete Performance and Availability monitoring Infrastructure with the goals of autonomic and on demand computing. TMTP supports these goals by including the following integration points.

- IBM Tivoli Monitoring (ITM): ITM provides monitoring for system level resources to detect bottlenecks and potential problems and automatically recover from critical situations. This saves system administrators from manually scanning through extensive performance data before problems can be resolved. ITM incorporates industry best practices in order to provide immediate value to the enterprise. TMTP provides integration with ITM through the ability to launch the ITM Web Health Console in the context of a poorly performing transaction component (Figure 2-12). This is a powerful feature, as it allows you to drill down to a lower level from your poorly performing transaction and can allow you to identify issues such as poorly configured systems. Also with the addition of products such as IBM Tivoli Monitoring for Databases, IBM Tivoli Monitoring for Web Infrastructure, and IBM Tivoli Monitoring for Business Integration you will be further able to diagnose infrastructure problems and, in many cases, resolve them prior to their impacting the performance of your e-business transactions.

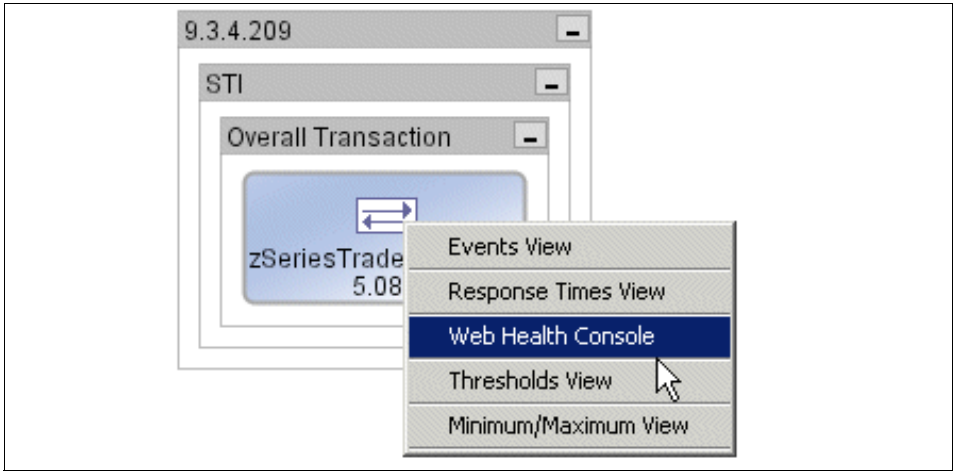


Figure 2-12 Launching the Web Health Console from the Topology view

- Tivoli Enterprise Console (TEC): The IBM Tivoli Enterprise Console provides sophisticated automated problem diagnosis and resolution in order to improve system performance and reduce support costs. Any events generated by TMTP can be automatically forwarded to the TEC. TMTP ships with the Event Classes and rules for TEC to make use of event information from TMTP.
- Tivoli Data Warehouse (TDW): TMTP ships with both ETL1 and ETL2, which are required to use the Tivoli Data Warehouse. This allows historical TMTP data to be collected and analyzed. It also allows TMTP to be used with other Tivoli products, such as the Tivoli Service Level Advisor product. Chapter 10, “Historical reporting” on page 375 describes historical reporting for TMTP with the Tivoli Data Warehouse in some depth.
- Tivoli Business Systems Manager (TBSM): IBM Tivoli Business Systems Manager simplifies management of mission-critical e-business systems by providing the ability to manage real-time problems in the context of an enterprise's business priorities. Business systems typically span Web, client-server, and/or host environments, are comprised of many interconnected application components, and rely on diverse middleware, databases, and supporting platforms. Tivoli Business Systems Manager provides customers a single point of management and control for real-time operations for end-to-end business systems management. Tivoli Business Systems Manager enables you to graphically monitor and control interconnected business components and operating system resources from one single console and give a business context to management decisions. It helps users manage business systems by understanding and managing the dependencies between business systems components and their underlying infrastructure. TMTP can be integrated with TBSM using either the Tivoli Enterprise Console or via SNMP.
- Tivoli Service Level Adviser (TSLA): TSLA automatically analyzes service level agreements and evaluates compliance while using predictive analysis to help avoid service level violations. It provides graphical, business level reports via the Web to demonstrate the business value of IT. As described above, TMTP ships with the required ETLs needed for the Tivoli Service Level Advisor to utilize the information gathered by TMTP to create and monitor service level agreement compliance.
- Simple Network Management Protocol (SNMP) Support: For environments that do not have existing TEC implementations, or where the preference is to integrate using SNMP, TMTP has the ability to generate SNMP traps when thresholds are breached or to monitor TMTP itself.
- Simple Mail Transport Protocol (SMTP): TMTP is also able to generate e-mail messages to administrators when transaction thresholds are breached or when TMTP encounters some error condition.

- Scripts: Lastly, TMTP has the capability to run a script in response to a threshold violation or system event. The script is run at the Management Agent and could be used to perform some type of corrective action.

Configuring TMTP to integrate with these products is discussed in more depth in Chapter 5, “Interfaces to other management tools” on page 153.

3

IBM TMTP architecture

This chapter describes the following:

- High level architectural overview of IBM Tivoli Monitoring for Transaction Performance
- Detailed architecture for IBM Tivoli Monitoring for Transaction Performance Web Transaction Performance (WTP)
- Introduction to the components of WTP
- Discussion of the various technologies used by WTP
- Putting it all together to implement a transaction monitoring solution for your e-Business environment

© Copyright IBM Corp. 2003. All rights reserved.

3.1 Architecture overview

As discussed in Chapter 2, “IBM Tivoli Monitoring for Transaction Performance in brief” on page 37, IBM Tivoli Monitoring for Transaction Performance (hereafter referred to as TMTP) is an application designed to ease the capture of Transaction Performance information in a distributed environment. TMTP was first released in the mid 90s as two products: Tivoli Web Services Manager and Tivoli Application Performance Monitoring. These two products were designed to perform similar functions and were combined in 2001 into a single product, IBM Tivoli Monitoring for Transaction Performance. This heritage is still reflected today by the existence of two components of TMTP, the Enterprise Transaction Performance (ETP) and Web Transaction Performance (WTP) components. This release of TMTP blurs the distinction between the components and sets the stage for future releases where there will no longer be a distinction between ETP and WTP.

3.1.1 Web Transaction Performance

The IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance component is the area of the TMTP product where most changes have been introduced with Version 5.2. The basic architecture is shown in Figure 3-1 and elaborated on in further sections.

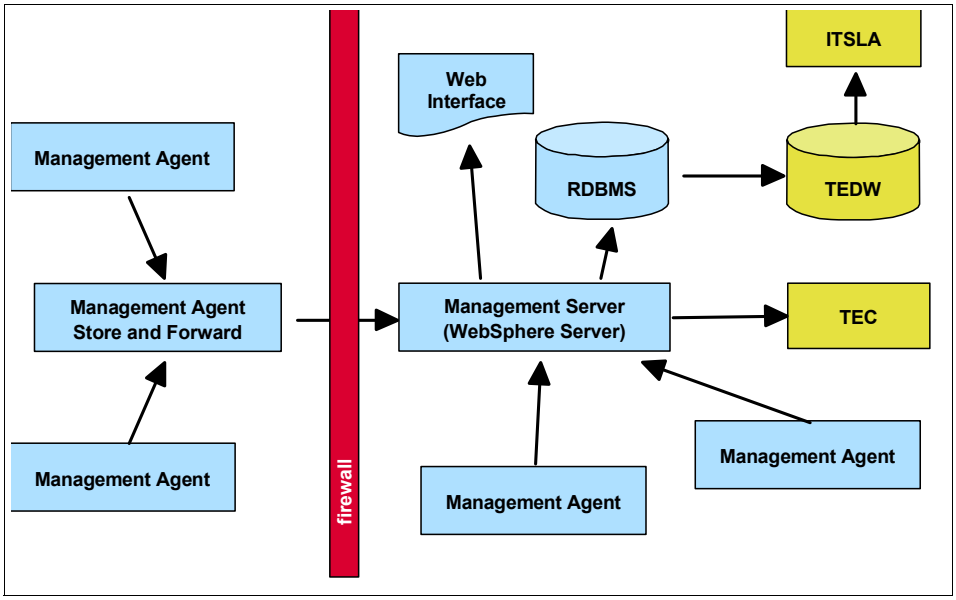


Figure 3-1 TMTP Version 5.2 architecture

This version of the product introduces a comprehensive transaction decomposition environment that allows users to visualize the path of problem transactions, isolate problems to their source, launch the IBM Tivoli Monitoring Web Health Console to repair the problem, and restore good response time.

WTP provides the following broad areas of functionality:

- Transaction definition

  The definition of a transaction is governed by the point at which it first comes in contact with the instrumentation available within this product. This can be considered the Edge definition, where each transaction, upon encountering the edge of the instrumentation available, will be defined through policies that define each transactions uniqueness specific to the Edge it encountered.

- Distributed transaction monitoring

  Once a transaction has been defined at its edge, there is a need for customers to define the policy that will be used in monitoring this transaction. This policy should control the monitoring of the transaction across all of the systems where it executes. To that end, monitoring policies are generic in nature and can be associated with any group of transactions.

- Cross system correlation

  One of the largest challenges in providing distributed Transaction Performance monitoring is the collection of subtransaction data across a range of systems for a specified transaction. To that end, TMTP uses an ARM correlator in order to correlate parent and child transactions.

All of the Web Transaction Performance components of ITM for TP share a common infrastructure based on the IBM WebSphere Application Server Version 5.0.1.

The first major component of Web Transaction Performance is the central Management Server and its database. The Management Server governs all activities in the Web Transaction Performance environment and controls the repository in which all objects and data related to Web Transaction Performance activity and use are stored.

The other major component is the Management Agent. The Management Agent provides the underlying communications mechanism and can have additional functionality implemented on to it.

The following four broad functions may be implemented on a Management Agent:

- Discovery: Enables automatic identification of incoming Web transactions that may need to be monitored.

- Listening: Provides two components that can “listen” to real end user transactions being performed against the Web servers. These components (also called listeners) are the *Quality of Service* and *J2EE monitoring* components.
- Playback: Provides two components that can robotically playback or execute transactions that have been recorded earlier in order to simulate actual user activity. These components are the *Synthetic Transaction Investigator* and *Rational Robot/Generic Windows* components.
- Store and Forward: May be implemented on one or more agents in your environment in order to handle firewall situations.

More details on each of these features can be found in 3.2, “Physical infrastructure components” on page 61.

3.1.2 Enterprise Transaction Performance

The Enterprise Transaction Performance (ETP) components are used to measure transaction performance from systems that belong to the Tivoli Management Environment. Typically, this implies that the transactions that are monitored take place between systems that are part of the enterprise network, also known as the intranet.

ETP has changed little, with the exception of the inclusion of the Rational Robot, since the previous version of ITM for TP and is only discussed briefly in this redbook. Other Redbooks that cover this topic more completely are:

- *Introducing Tivoli Application Performance Management*, SG24-5508
- *Tivoli Application Performance Management Version 2.0 and Beyond*, SG24-6048
- *Unveil Your e-business Transaction Performance with IBM TMTP 5.1*, SG24-6912

ETP provides four ways of measuring transaction performance:

- ARMed application
- Predefined Enterprise Probes
- Client Capture (browser-based)
- Record and Playback

However, the base technology used in probes, Client Capture, and Record and Playback is that of ARM; Enterprise Transaction Performance provides the means to capture and manage transaction performance data generated by ARM calls. It also provides a set of ARMed tools to facilitate data gathering and provide transaction performance data from applications that are not ARMed themselves.

Applications that are ARMed issue calls to the Application Response Measurement API to notify the ARM receiver (in this case implemented by Tivoli) about the specifics of the transactions within the application.

The probes are predefined ARMed programs provided by Tivoli that may be used to verify the availability of and the response time to load Web sites, mail servers, Lotus® Notes® Servers, and more. The specific object to be targeted by a probe is provided as run-time parameters to the probe itself.

Client Capture acts like a probe. When activated, it scans the input buffer of the browser of a monitored system (typically an end user's workstation) for specific patterns defined at the profile level and records the response time of all page loads, which matches the patterns specified.

The previous version of TMTP included two different implementations of transaction recording and playback: Mercury VuGen, which supports a standard browser interface, and the IBM Recording and Playback Workbench, which provides recording capabilities for 3270 and SAP transactions. This release of TMTP adds the Rational Robot as an enhanced mechanism for recording and playing back generic Windows transactions. The Rational Robot functionality applies to both the ETP and WTP components of TMTP, and is more completely integrated with the WTP component. Appendix B, "Using Rational Robot in the Tivoli Management Agent environment" on page 439 discusses ways of integrating the Rational Robot with the ETP component.

Figure 3-2 on page 60 gives an overview of the ETP architecture.

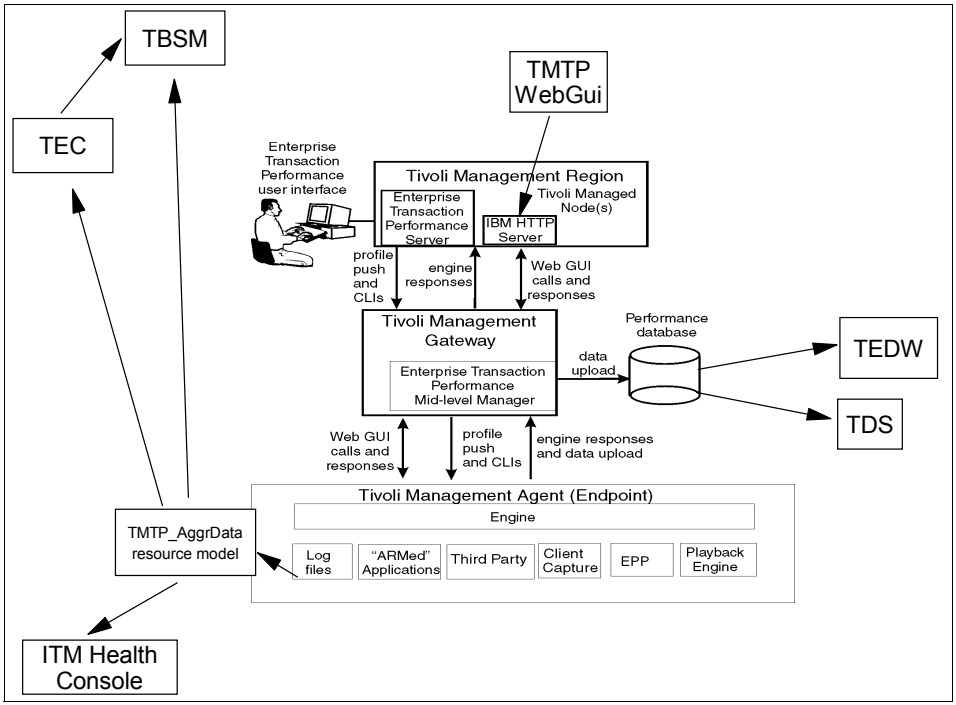


Figure 3-2 Enterprise Transaction Performance architecture

To initiate transaction performance monitoring, a MarProfile, which contains all the specifics of the transactions to be monitored, is defined in the scope of the Tivoli Management Framework and distributed to a Tivoli endpoint for execution. Based on the settings in the MarProfile, data is collected locally at the endpoint and may be aggregated to provide minimum, maximum, and average values over a preset period of time. Data related to specific runs of the transactions (instance data) and aggregated data may be forwarded to a central database, which may be used as the source for report generation through Tivoli Decision Support, and as data provider for other applications through Tivoli Enterprise Date Warehouse.

Online surveillance is facilitated through a Web-based console, on which current data at the endpoint and historical data from the database may be viewed.

In addition, two sets of monitors, a monitoring collection for Tivoli Distributed Monitoring 3.x and a resource model for IBM Tivoli Monitoring 5.1.1, are provided to enable generation of alerts to TEC and online surveillance through the IBM Tivoli Monitor Web Health Console. Note that both monitors are based on the aggregated data collected by the ARM receiver running at the endpoints and thus will not react immediately if, for example, a monitored Web site becomes

unavailable. The minimum time for reaction is related to the aggregation period and the thresholds specified.

3.2 Physical infrastructure components

As mentioned previously, all of the components of IBM Tivoli Monitoring for Transaction Performance share a common infrastructure based on the IBM WebSphere Application Server Version 5.0.1. This provides the TMTP product with a lot of flexibility. The TMTP Management Server is a J2EE application deployed onto the WebSphere Application Server platform. The installation of WebSphere and the deployment of the Management Server EAR are transparent to the installer. The Management Server provides the services and user interface needed for centralized management. Management agents are installed on computers across the environment. Management agents run discovery operations and collect performance data for monitored transactions. The Management Server and Management Agents may be deployed on the AIX®, Solaris, Windows, and xLinux platforms.

Another key feature of the IBM Tivoli Monitoring for Transaction Performance infrastructure is the application response measurement (ARM) engine. The ARM engine provides a set of interfaces that facilitate robust performance data collection.

The following sections describe the Management Server, Management Agents, and ARM in more detail.

The Management Server

The Management Server is shared by all IBM Tivoli Monitoring for Transaction Performance components and serves as the control center of your IBM Tivoli Monitoring for Transaction Performance installation. The Management Server collects information from, and provides services to, the Management Agents deployed in your environment. Management Server components are Java Management Extensions (JMX) MBeans.

Deployed as a standard WebSphere Version 5.0.1 EAR file, the Management Server provides the following functions:

- User interface: You can access the user interface provided by the Management Server through a Web browser running Internet Explorer 6 or higher. From the user interface, you create and schedule the policies that instruct monitoring components to collect performance data. You also use the user interface to establish acceptable performance metrics, or thresholds, define notifications for threshold violations and recoveries, view reports, view system events, manage schedules, and perform other management tasks.

- Real-time reports: Accessed through the user interface, real-time reports graphically display the performance data collected by the monitoring and playback components deployed in your environment. The reports enable you to quickly assess the performance and availability of your Web sites and Microsoft Windows applications.
- Event system: The Management Server notifies you in real time of the status of the transactions you are monitoring. Application events are generated when performance thresholds exceed or fall below acceptable limits. System events are generated for system errors and notifications. From the user interface, you can view recently generated events at any time. You can also configure event severities and indicate the actions to be taken when events are generated.
- Object model store for monitoring and playback policies: The object model store contains a set of database tables used to store policy information, events, and other information.
- ARM data persistence: All of the performance data collected by Management Agents is sent using the ARM API. The Management Server keeps a persistent record of the ARM data collected by Management Agents for use in real-time and historical reports.
- Communication with Management Agents: The Management Server uses Web services to communicate with the Management Agents in your environment.

Figure 3-3 gives an overview of the Management Server architecture.

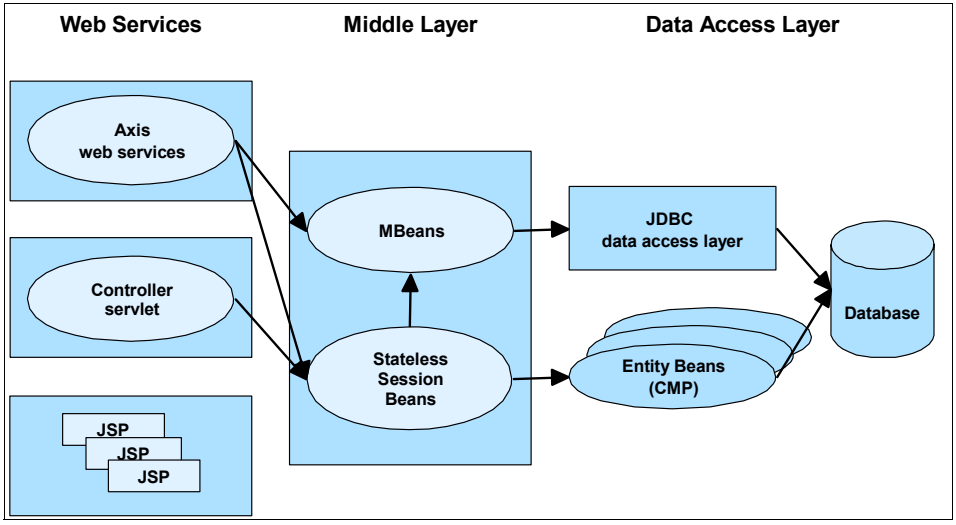


Figure 3-3 Management Server architecture

The Management Server components are JMX MBeans running on the MBeanServer provided by WebSphere Version 5.0.1. Communications between the Management Agents and the Management Server is via SOAP over HTTP or HTTPS (using a customized version of the Apache Axis 1.0 SOAP implementation) (see Figure 3-4). The services provided by the Management Server to the Management Agents are implemented as Web Services and invoked by the Management Agent using the Web Services Invocation Framework (WSIF). All downcalls from the Management Server to the Management Agent are remote MBean method invocations.

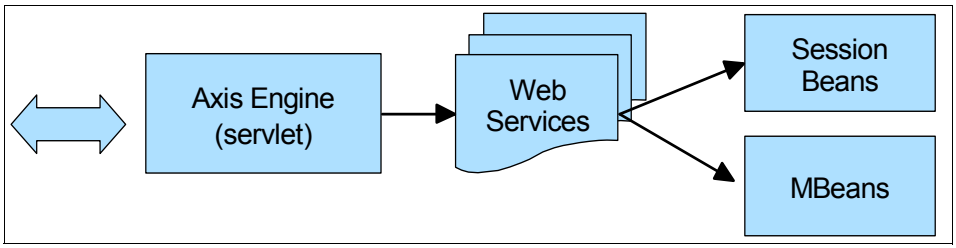


Figure 3-4 Requests from Management Agent to Management Server via SOAP

Note: The Management Sever application is a J2EE 1.3.1 application that is deployed as a standard EAR file (named tmtp52.ear). Some of the more important modules in the EAR file are:

- Report and User Interface Web Module: ru_tmtp.war
- Web Service Web Module: tmtp.war
- Policy Manager EJB Module: pm_ejb.jar
- User Interface Business Logic EJB Module: uiSessionModule.jar
- Core Business Logic EJB Module: sessionModule.jar
- Object Model EJB Module: entityModule.jar

ARM data is uploaded to the Management Server from Management Agents at regularly scheduled intervals (the upload interval). By default, the upload interval is once per hour.

The Management Agent

Management Agents are installed on computers across your environment. Based on Java Management Extensions (JMX), the Management Agent software provides the following functionality:

- Listening and playback behaviors: A Management Agent can have any or all of the listening and playback components installed. The components

associated with a Management Agent run policies at scheduled times. The Management Agent sends any events generated during a listening or playback operation to the Management Server, where event information is made available in event views and reports.

- ARM engine for data collection: A Management Agent uses the ARM API to collect performance data. Each of the listening and playback components is instrumented to retrieve the data using ARM standards.
- Policy management: When a discovery, listening, or playback policy is created, an agent group is assigned to run the policy. You define agent groups to include one or more Management Agents that are equipped to run the same policy. For example, if you want to monitor the performance of a consumer banking application that runs on several WebSphere application servers, each of which is associated with a Management Agent and a J2EE monitoring component, you can create an agent group named All J2EE Servers. All of the Management Agents in the group can run a J2EE listening policy that you create to monitor the banking application.
- Threshold setting: Management agents are capable of conducting a range of sophisticated threshold setting operations. You can set basic performance thresholds that generate events and send notification when a transaction exceeds or falls below an acceptable performance time. Other thresholds monitor for the existence of HTTP response codes or specified page content, or watch for transaction failure. In many cases, you can specify thresholds for the subtransactions of a transaction. A subtransaction is one step in the overall transaction.

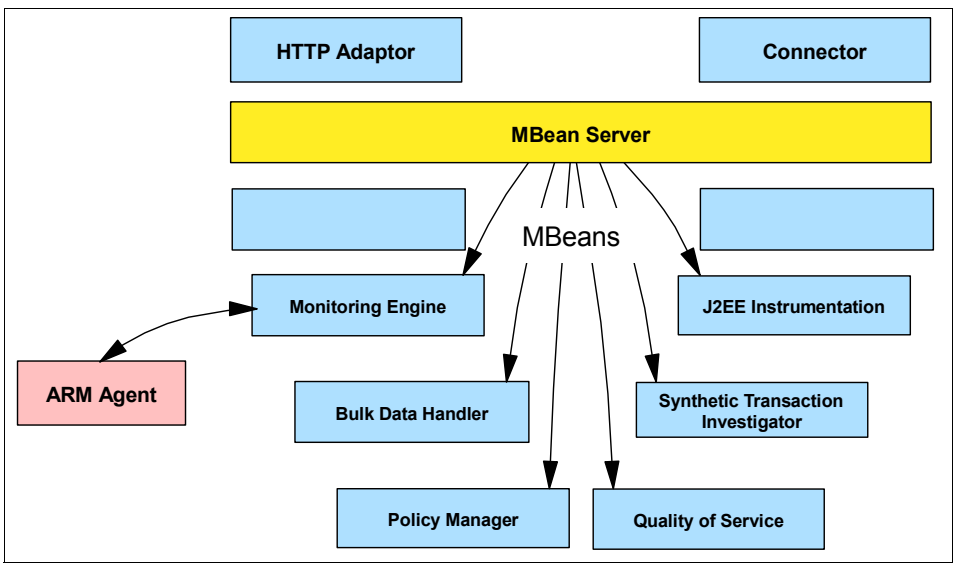


Figure 3-5 Management Agent JMX architecture

- Event support: Management agents send component events to the Management Server. A component event is generated when a specified performance constraint is exceeded or violated during a listening or playback operation. In addition to sending an event to the Management Server, a Management Agent can send e-mail notification to specified recipients, run a specified script, or forward selected event types to the Tivoli Enterprise Console or the simple network management protocol (SNMP).
- Communication with the Management Server: Management Agents communicate with the Management Server using Web services and the secure socket layer (SSL). Every 15 minutes, all Management Agents poll the Management Server for any new policy information (known as the polling interval).
- Store and Forward: Store and Forward can be implemented on one or more Management Agents in your environment (typically only one) to handle firewall situations. Store and Forward performs the following firewall-related tasks in your environment:
  - Enables point-to-point connections between Management Agents and the Management Server
  - Enables Management Agents to interact with Store and Forward as if Store and Forward were a Management Server
  - Routes requests and responses to the correct target
  - Supports SSL communications
  - Supports one-way communications through the firewall

All applications, such as STI, QoS, and J2EE, are registered as MBeans, as are all services used by the Management Agent and Server, for example, Scheduler, Monitoring engine, Bulk Data Transfer, and the Policy Manager service.

The Application Response Measurement Engine

When you install and configure a Management Agent in your environment, the Application Response Measurement (ARM) Engine is automatically installed as part of the Management Agent. The engine and ARM API comply with the ARM 2.0 specification. The ARM specification was developed in order to meet the challenge of tracking performance through complex, distributed computing networks. ARM provides a way for business applications to pass information about the subtransactions they initiate in response to service requests that flow across a network. This information can be used to calculate response times, identify subtransactions, and provide additional data to help you determine the cause of performance problems. Some of the specific details of how ARM is utilized by TMTP are discussed in the next section.

Figure 3-6 gives an overview of how the ARM Engine communicates with the Monitoring Engine.

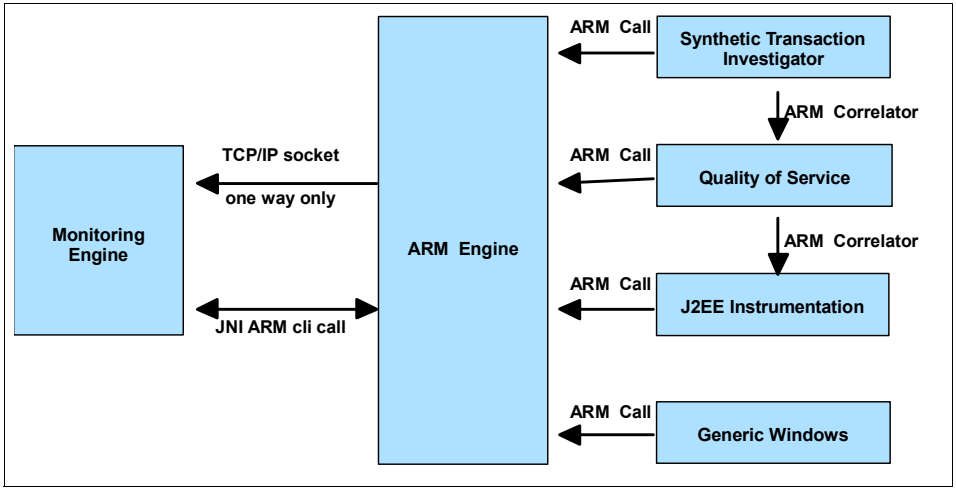


Figure 3-6 ARM Engine communication with Monitoring Engine

All transaction data collected by the Quality of Service, J2EE, STI, and Generic Windows monitoring components of TMTP is collected by the ARM functionality. The use of ARM results in the following capabilities:

- Data aggregation and correlation: ARM provides the ability to average all of the response times collected by a policy, a process known as aggregation. Response times are aggregated once per hour. *Aggregate data* gives you a view into the overall performance of a transaction during a given one-hour period. *Correlation* is the process of tracking hierarchical relationships among transactions and associating transactions with their nested subtransactions. When you know the parent-child relationships among transactions and the response times for each transaction, you are much better able to determine which transactions are delaying other transactions. You can then take steps to improve the response times of services or transactions that contribute the most to slow performance.
- Instance and aggregate data collection: When a policy collects performance data, the collected data is written to disk. Because Management Agents are equipped with ARM functionality, you can specify that aggregate data only be written to disk (to conserve system resources and view fewer data points) or that both aggregate and instance data be written to disk. Aggregate data is an average of all response times detected by a policy over a one-hour period, whereas instance data consists of response times that are collected every time the transaction is detected. TMTP will normally collect only aggregate data unless instance data collection was specified in the listening policy.

TMTP will also automatically collect instance data if a transaction breaches specified thresholds. This second feature of TMTP is very useful, as it means that TMTP does not have to keep redundant instance data, yet has relevant instance data should a transaction problem be recognized.

3.3 Key technologies utilized by WTP

This section describes some of the technologies used in this release of TMTP and elaborates on some of the changes introduced to how some previously implemented technologies are utilized.

3.3.1 ARM

The Application Response Measurement (ARM) API is the key technology utilized by TMTP to capture transaction performance information. The ARM standard describes a common method for integrating enterprise applications as manageable entities. It allows users to extend their enterprise management tools directly to applications, creating a comprehensive end-to-end management capability that includes measuring application availability, application performance, application usage, and end-to-end transaction response time. The ARM API defines a small set of functions that can be used to instrument an application in order to identify the start and stop of important transactions. TMTP provides an ARM engine in order to collect the data from ARM instrumented applications.

The ARM standard has been utilized by several releases of TMTP, so it will not be discussed in great depth here. If the reader wishes to explore ARM in detail, the authors recommend the following Redbooks, as well as the ARM standard documents maintained by the Open Source Group (available at http://www.opengroup.org):

- *Introducing Tivoli Application Performance Management,* SG24-5508
- *Tivoli Application Performance Management Version 2.0 and Beyond,* SG24-6048
- *Unveil Your e-business Transaction Performance with IBM TMTP 5.1,* SG24-6912

The TMTP ARM engine is a multithreaded application implemented as the tapmagent (tapmagent.exe on Windows based platforms). The ARM engine exchanges data though an IPC channel, using the libarm library (libarm32.dll on Windows based platforms), with ARM instrumented applications. The data collected is then aggregated in order to generate useful information, correlated with other transactions, and thresholds are measured based upon user

requirements. This information is then rolled up to the Management Server and placed into the database for reporting purposes.

The majority of the changes to the ARM Engine pertain to measurement of transactions. In the TMTP 5.1 version of the ARM Engine, each and every transaction was measured for either aggregate information or instance data. In this version of this component, the Engine will be notified as to which transactions need to be measured. This is done via new APIs to the ARM Engine that allows callers to identify transactions, either explicitly or as a pattern. Measurement can be defined for “edge” transactions, which will result in response measurement of the edge and all its subtransactions.

Another large change in the functionality of the ARM Engine is monitoring for threshold violations of a given transaction. Once a transaction is defined to be measured by the ARM Engine, it can also be defined to be monitored for threshold violations. A threshold violation is defined in this release of this component to be completing the transaction (i.e. arm_stop) and having a unsuccessful return code or having a duration greater than a MAX threshold or less than a MIN threshold.

The ARM Engine will also communicate with the Monitoring Engine to inform it of transaction violations, new edge transactions appearing, and edge transaction status changes.

ARM correlation

ARM correlation is the method by which parent transactions are mapped to their respective child transactions across multiple processes and multiple servers.

This release of the TMTP WTP component provides far greater automatic support for the ARM correlator. Each of the components of WTP is automatically ARM instrumented and will generate a correlator. The initial root/parent or “edge” transaction will be the only transaction that does not have a parent correlator. From there, WTP can automatically connect parent correlators with child correlators in order to trace the path of a distributed transaction through the infrastructure and provides the mechanisms to easily visualize this via the topology views. This is a great step forward from previous versions of TMTP, where it was possible to generate the correlator, but the visualization was not an automatic process and could be quite difficult.

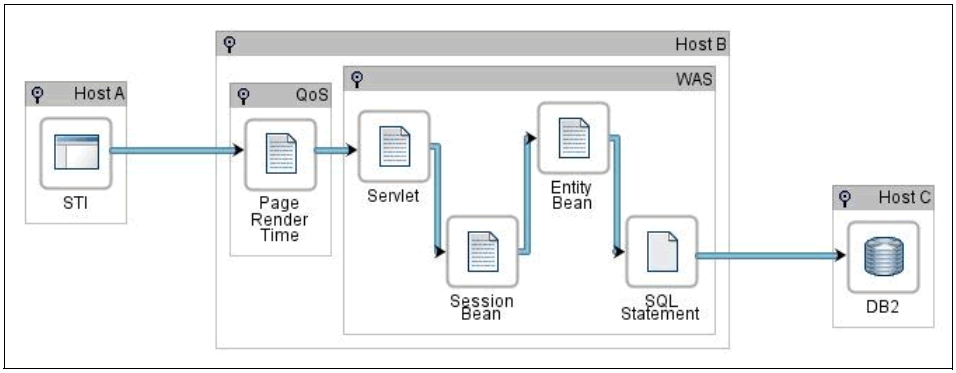


Figure 3-7 Transaction performance visualization

TMTP Version 5.2 implements the following ARM correlation mechanisms:

1. Parent based aggregation

   Probably the single largest change to the current ARM aggregation agent is the implementation of parent based correlation. This enables transaction performance data to be collected based on the parent of a subtransaction. This allows the displaying of transaction performance relative to its path. The purpose served by this is the ability to monitor the connection points between transactions. It also enables path based transaction performance monitoring across farms of servers all providing the same functionality. The correlator generation mechanism will pass parent identification within the correlator to enable this to occur.

2. Policy based correlators

   Another change for the correlator is that a portion of the correlator is used to pass a unique policy identifier within the correlator. The associated policy will control the amount of data being collected and also the thresholds associated with that data. In this model, a user specifies the amount of data collection for the different systems being monitored. Users do not need to know the actual path taken by a transaction and can accept the defaults in order to achieve an acceptable level of monitoring. For specific transactions, users can create unique policies that provide a finer level of control over the monitoring of those transactions. An example would be the decision to enable subtransaction collection of all methods within WebSphere, as opposed to the default of collecting only Servlet, EJB, JMS, and JDBC.

3. Instance and aggregated performance statistics

   Users have come to expect support for the collection of instance performance data. This provides both additional metrics and a complete and exact trace of the path taken by a specific transaction. The TMTP 5.1 ARM agent implementation was designed to provide an either/or model where all

statistics are collected as instance or aggregate, regardless of the specific transaction being monitored. Support is provided by TMTP Version 5.2 for collecting both instance and aggregate at the same time. All ARM calls contain metrics, regardless of the users request to store instance data. This occurs because the application instrumentation is unaware of any configuration selections made at higher levels. In the past, the ARM agent, when collecting aggregated data, would normally discard the metric data provided to it. This has been changed so that any ARM call that becomes the MAX for a given aggregation period will have its metrics stored and maintained. This functionality enables a user to view the context (metrics) associated with the worst performing transaction for a given time period. It is important to note (see parent based aggregation) that the term “worst performing” is specific to each subtransaction individually and not the overall performance of the parent transaction. However, the MAX for each subtransaction within a given transaction will store its context uniquely, allowing for the presentation of the complete transaction, including the context of each subtransaction performing at its own worst level.

4. Parent Performance Initiated Trace

   The trace flag within the ARM correlator is utilized by the agent (x'80' in the trace field) for transactions that are performing outside of their threshold. This provides for the dynamic collection of instance data across all systems where this transaction executes. The ARM agent at the transaction initiating point enables this flag when providing a correlator for a transaction that has performed slower then its specified threshold. To limit the overall performance impact of this tracing, this flag is only generated once for each transaction threshold crossing. Trace will continue to be enabled for this transaction for up to five consecutive times unless transaction performance recedes below threshold. This should enable the tracing of instance data for a violating transaction without user intervention, while allowing for aggregated collection of data at all other times. For the unique cases where these violations are not caught via this mechanism, it is expected that a user will change the monitoring policy for this transaction to be an instance in order to ensure the capture of an offending transaction. Given that each MAX transaction (and subtransaction) will already have instance metrics, the benefits of this will be seen in the collection of subtransactions that were normally not being traced. The last statement is due to the fact that a monitoring policy may preclude the collection of all subtransactions within WebSphere (and possibly other applications) from occurring during normal monitoring. To enable a complete breakdown of the transaction, all instrumentation agents collect all data when the trace flag is present.

5. Sibling transaction ordering

   Sibling transaction ordering is the ability to determine the order of execution of a set of child transactions relative to each other. However, when ordering

sibling transactions from data collected across multiple systems, the information gathered may not be entirely correct because of time synchronization issues. In case the system clocks on all the machines involved are *not* synchronized, the recorded data may show sibling transaction ordering sequences that are not entirely correct. This will not affect the overall flow of the transaction, only the presentation of the ordering of child transactions in situations where the child transactions execute on different systems. The recommendation is to synchronize the system clocks if you are concerned about the presentation of sibling transaction ordering.

This release of TMTP adds the notion of aggregated correlation. Aggregated correlation will provide aggregate information (that is, does not create a record for each and every instance of a transaction, but a summary of a transaction over a period of time). Instead of a singular transaction being aggregated, correlation will be used. Previous versions of TMTP only allowed correlation at the instance level, which could be an intensive process.

The logging of transactions will usually start out as aggregated correlation. There may be times when a registered measurement entry will be provided to the ARM Engine that will ask for instance logging, or the ARM Engine itself may turn on instance logging in the event of a threshold violation.

There are essentially three ways TMTP treats aggregated correlation:

1. Edge aggregation by pattern
2. Edge aggregation by transaction name (edge discovery mode)
3. Aggregation by root/parent/transaction

For *edge aggregation by pattern*, we essentially have one aggregator per edge policy that all transactions that match that edge policy pattern will be aggregated against.

For *edge aggregation by transaction name*, we essentially have a unique aggregator for each transaction name that matches this policy's edge pattern. This is what we deem discovery mode, because in this situation, we will be "discovering" all the edges that match the specified edge pattern. When in discovery mode, TMTP always generates a correlator with the TMTP_Flags ignore flag set to true to signal that we do not want to process subtransactions.

For all non-edge aggregation, we will be performing correlated aggregation. What this means is each transaction instance will be directed to a specific aggregator based upon correlation using the following four properties:

1. Origin host UUID
2. Root transID
3. Parent transID
4. Transaction classID

By providing this correlation information in the aggregation, you are better able to see the aggregation information in respect to the code flow of the transactions that have run.

Every hour, on the hour, this information will be sent to an outboard file for upload to the Management Server Database.

How are correlators passed from one component to the next?

Each component of TMTP passes the correlator it has generated to each of its subtransactions using Java RMI over IIOP. Java RMI over IIOP combines Java Remote Method Invocation (RMI) technology with Internet Inter-Orb Protocol (IIOP - CORBA technology) and allows developers to pass any serialized Java object (Objects By Value) between application components.

Transactions entering the J2EE Application Server may already have a correlator associated, which has been generated because the transaction is being monitored by one of the other TMTP components, such as QoS, STI, J2EE instrumentation on another J2EE Application Server, or Rational/Genwin. If no correlator exists when a transaction enters the J2EE Application Server, the server:

- Requests a correlator from ARM.
- If no policy matches, J2EE does not get a correlator.
- Subtransactions can detect their parent correlator.
- If no correlator, performance data is not collected.
- If correlator, performance data is logged.

In summary

This version of TMTP uses parent based aggregation where subtransactions are chained together based on correlators, allowing TMTP to generate the call stack (transaction path). The aggregation is policy based, which means that information is only collected for transactions that match the defined policy. Additionally, TMTP will dynamically collect instance data (as opposed to aggregated data) based on threshold violations. TMTP also allows child subtransactions to be ordered based on start times.

3.3.2 J2EE instrumentation

In this section we describe one of the key enhancements included with the release of TMTP Version 5.2: its ability to do J2EE monitoring at the subtransaction level without the use of manual instrumentation.

The problem

There are many applications written in J2EE that are hosted on various different J2EE application servers at varying version levels. A J2EE transaction can be made up of many components, for example, JSPs, Servlets, EJBs, JDBC, and so on. This level of complexity makes it hard to identify if there is a problem and where that problem lies. We need a mechanism for finding the component that is causing the problem.

J2EE support provided by TMTP 5.1

In TMTP 5.1, the ETP component could collect ARM data generated by applications on WebSphere servers that had IBM WebSphere Application Server Version 5.0 installed. This data was provided by the WebSphere Request Metrics facility.

This was a start, but only limited detail was provided, such as the number of servlets and number of EJBs. The ETP component could supplement this data by collecting ARM data independently of the STI Player and the STI player could trigger the collection of ARM data on its behalf.

ETP then uploaded all the ARM data from all the transactions within an application that have been configured in WebSphere. The administrator could turn data collection on or off at the application level.

These capabilities solved some business problems, but led to the need for greater control and granularity, as well as the need for greater scope.

J2EE support provided by TMTP Version 5.2

TMTP Version 5.2 provides enhanced J2EE instrumentation capabilities. The collection of ARM data generated by J2EE applications is invoked from the new Management Server, not from ETP. The ARM collection is controlled by user configured policies that are created on the Management Server. The process of creating appropriate J2EE discovery and listening policies is described in Chapter 8, “Measuring e-business transaction response times” on page 225. The monitoring policy is then distributed to the Management Agent.

Which transactions to monitor are specified using edge definitions, for example, the first URI invoked when utilizing the application, and it is possible to define the level of monitoring for each edge.

In order to monitor a J2EE Application Server, the machine must be running the TMTP Agent. A single TMTP agent can monitor multiple J2EE Application Servers on the Management Agent’s host.

TMTP Version 5.2 provides J2EE monitoring for the following J2EE Application Servers:

- WebSphere Application Server 4.0.3 Enterprise Edition and later
- BEA WebLogic 7.0.1

TMTP's J2EE monitoring is provided by Just In Time Instrumentation (JITI). JITI allows TMTP to manage J2EE applications that do not provide system management instrumentation by injecting probes at class-load time, that is, no application source code is required or modified in order to perform monitoring. This is a key differentiator between TMTP and other products, which can require large changes to application source code. Additionally, the probes can easily be turned on and off as required. This is an important difference, which means that the additional transaction decomposition can be turned on only when required. It is important that this capability is available as though TMTP has low overheads (all performance monitoring has some overhead; the more monitoring you do the greater the overhead). The fact that J2EE monitoring can be easily enabled and disabled based on a policy request from the user is a powerful feature.

Just In Time Instrumentation explained

As discussed above, one of the key changes introduced by this release of ITM for TP is the introduction of Just In Time Instrumentation (hereafter referred to as JITI). JITI builds on the performance "listening" capabilities provided in previous versions by the QoS component to allow detailed performance data to be collected for J2EE (Java 2 Platform Enterprise Edition) applications without requiring manual instrumentation of the application.

How it works

With the release of JDK 1.2, Sun included a profiling mechanism within the JVM. This mechanism provided an API that could be used to build profilers called JVMPI, or Java Virtual Machine Profiling Interface. The JVMPI is a bidirectional interface between a Java virtual machine and an in-process profiler agent. JITI uses the JVMPI and works with un-instrumented applications.

The JVM can notify the profiler agent of various events, corresponding to, for example, heap allocation, thread start, and so on. Or the profiler agent can issue controls and requests for more information through the JVMPI, for example, the profiler agent can turn on/off a specific event notification, based on the needs of the profiler front end.

As shown by Figure 3-8 on page 75, JITI starts when the application classes are loaded by the JVM (for example, the WebSphere Application Server). The Injector alters the Java methods and constructors specified in the registry by injecting special byte-codes in the in-memory application class files. These byte-codes include invocations to hook methods that contain the logic to manage

the execution of the probes. When a hook is executed, it gets the list of probes currently enabled for its location from the registry and executes them.

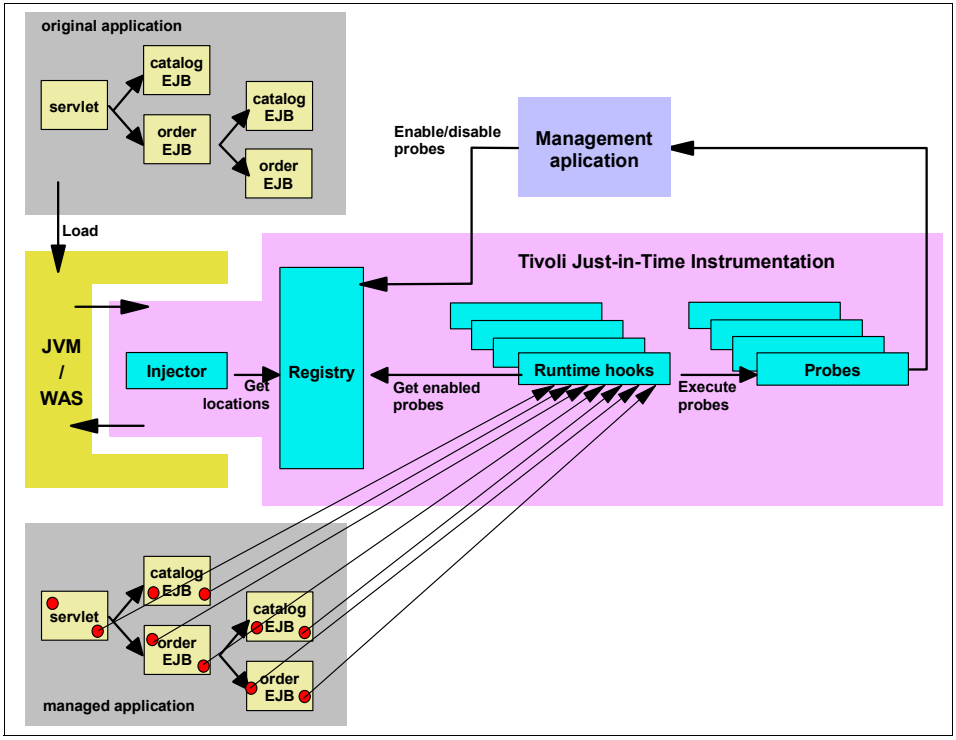


Figure 3-8 Tivoli Just-in-Time Instrumentation overview

TMTP Version 5.2 bundles JITI probes for:

- Servlets (also includes Filters, JSPs)
- Entity Beans
- Session Beans
- JMS
- JDBC
- RMI-IIOP

JITI combined with the other mechanisms included with TMTP Version 5.2 allow you to reconstruct and follow the path of the entire J2EE transaction through the enterprise.

TMTP J2EE monitoring collects instance level metric data at numerous locations along the transaction path. Servlet Metric Data includes URI, querystring, parameters, remote host, remote user, and so on. EJB Metric Data includes

primary key, EJB type (stateful, stateless, and entity), and so on. JDBC Metric Data includes SQL statement, remote database host, and so on.

JITI probes make ARM calls and generates correlators in order to allow subtransactions to be correlated with their parent transactions.

The primary or root transaction is the transaction that has no parent correlator and indicates the first contact of the transaction with TMTP. Each transaction monitored with TMTP gets its own correlator, as does each subtransaction. When a subtransaction is started, ARM can link it with its parent transaction based on the correlators and so on down the tree. With the correlator information, ARM can build the call tree for the entire transaction.

If a transaction crosses J2EE Application Servers on multiple hosts, the ARM data can be captured by installing the Management Agent on each of the hosts. Only the host that registers the root transaction need have a J2EE Listening Policy.

TMTP Version 5.2 J2EE monitoring summarized

- JITI provides the ability to monitor the fine details of any J2EE applications. It does this by dynamically inserting probes at run time.
- There is no need to re-run a command after deploying a new application.
- You can view a transaction path in Topology.
- It is easy to discover the root cause of a performance problem.
- You can discover new transactions you were not aware of in your environment.
- You can dynamically configure tracing details.
- You can run monitoring at a low trace level during normal operation.
- You can increase to a high tracing level after a problem is detected.

3.4 Security features

TMTP Version 5.2 includes features to allow your transaction monitoring infrastructure to be secure. The key features that support secure implementations are shown in the following sections.

SSL communications between components

SSL is a security protocol that provides for authentication, integrity, and confidentiality. Each of the components of TMTP Version 5.2 WTP can optionally be configured to utilize SSL for communications.

A sample HTTP-based SSL transaction using server-side certificates follows:

1. The client requests a secure session with the server.
2. The server provides a certificate, its public key, and a list of its ciphers to the client.
3. The client uses the certificate to authenticate the server (that is, to verify that the server is who they claim to be).
4. The client picks the strongest cipher that they have in common and uses the server's public key to encrypt a newly-generated session key.
5. The server decrypts the session key with its private key.
6. Henceforth, the client and server use the session key to encrypt all messages.

TMTP uses the Java Secure Sockets Extensions (JSSE) API to create SSL sockets within Java applications and includes IBM's GSKIT to manage certificates. Chapter 4, "TMTP WTP Version 5.2 installation and deployment" on page 85 includes information on how to configure the environment to use SSL.

Store and Forward Agent

The Store and Forward Management Service is a new component in the TMTP infrastructure. The service resides on a TMTP Management Agent. The new service was created in order to allow the TMTP Version 5.2 Management Server to be moved from the DMZ into the Enterprise. The agent enables a point-to-point connection between the TMTP Management Agents in the DMZ with the TMTP Management Server in the Enterprise. The functions provided by the Store and Forward agent (hereafter referred to as the SnF agent) are:

- Behaves as a pipe between the TMTP Management Server and TMTP Management Agents
- Maintains a single open and optionally persistent connection to the Management Server in order to forward agent requests
- Minimizes access from the DMZ through the firewall (one port for a SnF agent)
- Acts as part of the TMTP framework (that is, the JMX environment, User Interface, Policy, and so on).

Configuration of the SnF agent, including how to configure SnF to relay across multiple DMZs, is discussed further in Chapter 4, "TMTP WTP Version 5.2 installation and deployment" on page 85.

The SnF agent is comprised of two parts: the reverse proxy component, which utilizes WebSphere Caching Proxy, and the JMX TMTP agent, which manages the reverse proxy (both of these components will be installed transparently when

you install the SnF agent). The TMTP architecture, utilizing a SnF, precludes direct connection from the Management Server. All endpoint requests are driven to the Management Server via the reverse proxy. All communication between the SnF agent and the Management Server is via HTTP/HTTPS over a persistent connection. Connections to other Management Agents from the SnF agent are not persistent and are optionally SSL. The SnF agent performs no authorization of other Management Agents, as the TMTP endpoint is considered trusted, because registration occurs as part of a user/manual process.

Figure 3-9 shows the SnF Agent communication flows.

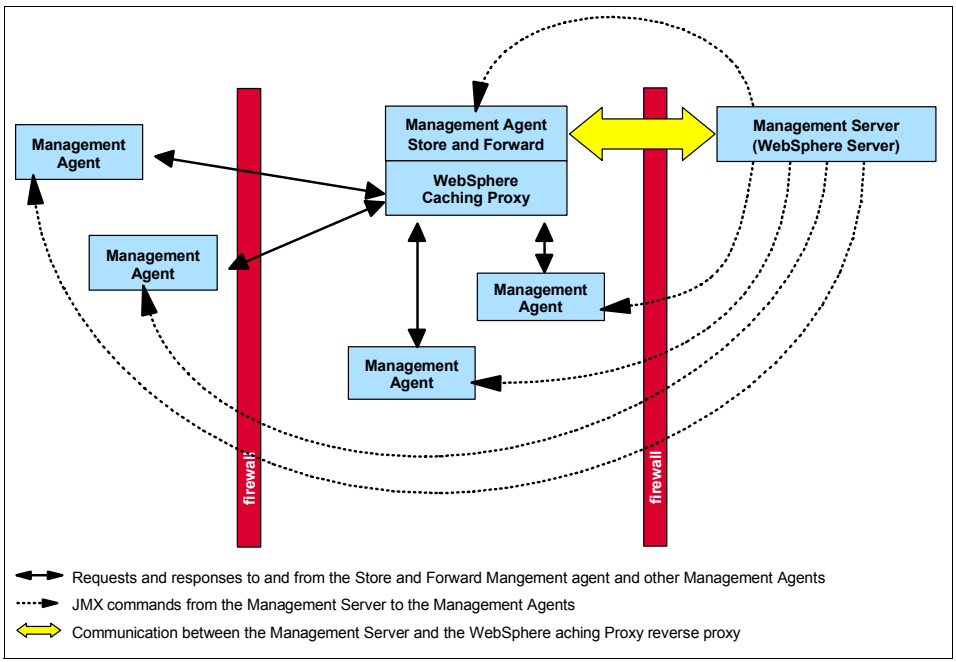

Management Agent
Management Agent Store and Forward
WebSphere Caching Proxy
Management Server (WebSphere Server)
Management Agent
Management Agent
Management Agent
firewall
firewall
Requests and responses to and from the Store and Forward Mangement agent and other Management Agents
JMX commands from the Management Server to the Management Agents
Communication between the Management Server and the WebSphere aching Proxy reverse proxy

Figure 3-9 SnF Agent communication flows

Ports used

Because of the Store and Forward agent, the number of ports used to communicate from the Management Agent to the Management Servers can be limited to one and communications via this port is secured using SSL. Additionally, each of the ports that are used by TMTP for communication between the various components can be configured. The default port usage and configuration of non default ports is discussed in Chapter 4, “TMTP WTP Version 5.2 installation and deployment” on page 85.

TMTP users and roles

TMTP uses WebSphere Application Server 5.0 security. This means that TMTP authentication can be performed using the operating system, that is, standard operating system user accounts, LDAP, or a custom registry. Also, the TMTP Application defines over 20 roles, which can be assigned to TMTP users in order to limit their access to the various functions which TMTP offers. Users are mapped to TMTP roles utilizing standard WebSphere Application Server 5.0 functionality. The process of mapping users to roles within WebSphere is described in Chapter 4, "TMTP WTP Version 5.2 installation and deployment" on page 85. Also, as TMTP uses WebSphere Security, it is possible to configure TMTP for Single Sign On (the details of how to do this are beyond the scope of this redbook; however, the documentation that comes with WebSphere 5.0.1 discusses this in some depth). The redbook *IBM WebSphere V5.0 Security*, SG24-6573 is also a useful reference for learning about WebSphere 5.0 security.

3.5 TMTP implementation considerations

Every organization's transaction monitoring requirements are different, which means that no two TMTP implementations will be exactly the same. However, there are several key considerations that must be made.

Where to place the Management Server

Previous versions of TMTP made this decision for you, as placing the Management Server (previously called TIMS) anywhere other than in the DMZ necessitated opening excessive additional incoming ports through your firewall. This release of TMTP includes the Store and Forward agent, which allows communications from the Management Agents to the Management Server to be consolidated and passed through a firewall via a single configured port. The Store and Forward agent can also be chained in order to facilitate communicate through multiple firewalls in a secure way. In general, the placement of the Management Server will be in a secure zone, such as the intranet.

Where to place Store and Forward agents

SnF agents can be placed within each DMZ in order to allow communications with the Management Server. By default, the SnF agent communicates directly with the Management Server; however, should your security infrastructure necessitate it, it is possible to use the SnF agent in order to connect multiple DMZs. This configuration is discussed in Chapter 4, "TMTP WTP Version 5.2 installation and deployment" on page 85.

Where and why to place QoSs

Placement of the QoS component is usually dictated by the placement of your Web Application Infrastructure Components. The QoS sits in front of your Web

server as a reverse proxy that forwards requests to the original Web server and relays the results back to the end user's Web browser. Several options are possible, such as in front of your load balancer, behind your load balancer, and on the same machine as your Web server. There is no hard and fast rule about the placement, so placement is dictated by what you want to measure. However, the QoS component is designed as a sampling tool. This means that in a large scale environment, where you have a Web Server farm behind load balancers, the QoS only needs to be in the path of one of your Web Servers. This will generally get a statistically sound sample that can be used to extrapolate the performance of your overall infrastructure.

Where and why to place the Rational/GenWin component

The GenWin component allows you to playback recorded transactions against generic Windows applications. Placement of the GenWin component will depend on what performance information you are trying to obtain and against what type of application you are trying to collect this information. If the application you are trying to capture end-user experience information for is an enterprise application, such as SAP or 3270, then the GenWin component will be placed within the intranet. However, if you are using the GenWin component to capture end-user experiences of your e-business infrastructure, it may make sense to place the GenWin component on the Internet. In general, STI is a better choice for capturing Internet-based transaction performance information, but in some cases, it may be unable to get the information that you require. A comparison of when and why to use GenWin versus STI is included in 8.1.2, "Choosing the right measurement component(s)" on page 229.

Where and Why to place STIs

The STI Management Agent is used to playback recorded STI scripts. Placement of the STI component is dictated by similar considerations as those used to decide where the GenWin component should be placed, that is, what performance data you are interested in and what application are you monitoring. If you are interested in capturing end-user experience data as close as possible to that experienced by users from the Internet or from partner organizations, you would place the STI component on the Internet or even within your partner organization. If this is of less interest, for example, if you are more interested in generating availability information, it may make sense to place the STI endpoint within the DMZ. Some of these considerations are discussed further in Chapter 8, "Measuring e-business transaction response times" on page 225.

3.6 Putting it all together

Figure 3-10 on page 81 shows a typical modern e-business application architecture around which we have placed the TMTP WTP components. This will

help the reader to visualize how the WTP components could be placed. The application architecture introduced below will form the basis of most of the scenarios that we cover in later chapters. In the rest of this book, we have used the Trade and PetStore J2EE applications for our monitoring scenarios. Each of these examples is shipped with WebSphere 5.0.1 and Weblogic. Figure 3-10 shows an e-business architecture that may be used to provide a highly scalable implementation of each of these applications.

Typical features of such an infrastructure include the use of a Web tier consisting of many Web servers serving up the applications static content and an Application tier serving up the dynamic content. Generally, a load balancer will be used by the Web tier to distribute application requests among the Web servers. Each Web Server may then use a plug-in to direct any requests for dynamic content from the Web Server to the back-end application server.

The application server provides many services to the application running on it, including data persistence, that is, access to back-end databases, access to messaging infrastructures, security, and possibly access to legacy systems.

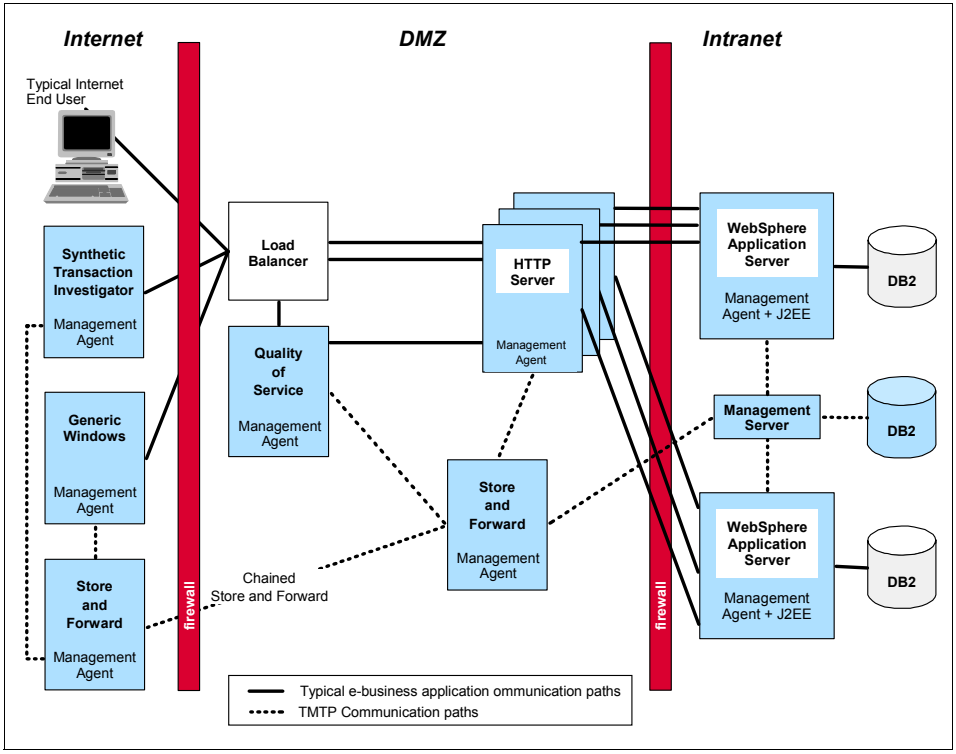


Figure 3-10 Putting it all together

In the design shown in Figure 3-10 on page 81, we have made the following placement decisions:

Management Server: We have placed it in the intranet zone, as this is the preferred and most secure location for the Management Server.

Store and Forward Management Agent: We have used only one and placed it in the DMZ. This will allow the Management Agents within the DMZ and on the Internet to securely communicate with the Management Server. Many environments may have multiple levels of DMZ, in which case chaining Store and Forward agents would have been a better option.

Quality of Service Management Agent: We have chosen to use only one and place it behind our load balance, yet in front of one of the back-end Web Servers. We considered that this solution would give us a good enough statistical sample to monitor end-user experience time. Another option which we considered seriously was placement of a Management Agent and Quality of Service endpoint on each of our Web Servers. This would have given us the capability to sample 100% of our traffic. We discarded this option, as we felt that we did not need this level of detail to satisfy our requirements.

Synthetic Transaction Investigator Management Agent: We chose to place one of these on the Internet, as this will allow us to closely simulate a real end user accessing our e-business transactions. We also plan to place additional Synthetic Transaction Investigator Management Agents both in the DMZ and intranet, as well as on the Internet as specific e-business transaction monitoring requirements arise.

Rational Robot/GenWin Management Agent: Again, we chose to place one of these on the Internet in order to allow us to test end-user response times of our e-business infrastructure where it uses Java applets or other content, which is not supported by the STI Management Agent. Later plans are to deploy Rational Robot/GenWin Management Agents within the enterprise in order to monitor the transaction performance of our other enterprise systems, such as SAP, Seibel, and our 3270 applications, from an end user's perspective.

J2EE Monitoring Management Agent: We chose to deploy the Management Agent and J2EE monitoring behavior to each of our WebSphere Web Application servers. This will provide us with the ability to do detailed transaction decomposition to the method level for our J2EE based applications.

Part 2

Installation and deployment

This part discusses issues related to the installation and deployment of IBM Tivoli Monitoring for Transaction Performance Version 5.2. In addition, information regarding the maintenance of the TMTP solution is provided. The following main topics are included:

- Chapter 4, “TMTP WTP Version 5.2 installation and deployment” on page 85
- Chapter 5, “Interfaces to other management tools” on page 153
- Chapter 6, “Keeping the transaction monitoring environment fit” on page 177

The target audience for this part is individuals who will plan for and perform an installation of IBM Tivoli Monitoring for Transaction Performance Version 5.2, as well as those who are responsible for the overall well-being of the transaction monitoring environment.

© Copyright IBM Corp. 2003. All rights reserved.

4

TMTP WTP Version 5.2 installation and deployment

In the first part of this chapter, we will demonstrate the installation of TMTP Version 5.2 in a production environment. There are two approaches to installing the TMTP Version 5.2 Management Server.

The first one is called “typical” installation, where the setup program will install and configure everything for you, including the required DB2® Version 8.1, WebSphere Application Server Version 5.0, and WebSphere Application Server FixPack 1.

The second approach is to install TMTP Version 5.2 in an environment where either the DB2 or the WebSphere Application Server or both are already deployed. This is called “custom” installation.

Both approaches have secure and a nonsecure options

We will use the custom secure installation option on AIX Version 4.3.3 in this scenario. We will show you how to configure your environment and how to prepare the previously installed DB2 Version 8.1 and WebSphere Version 5.0.1 Server to be able to install TMTP Version 5.2 smoothly. The description of this environment and the architecture can be found in 3.6, “Putting it all together” on page 80.

© Copyright IBM Corp. 2003. All rights reserved.

In the second part of this chapter, we will demonstrate a typical nonsecure installation suitable for the quick setup of the TMTP in a test or small business environment. SuSE Linux 7.3 will be used as an installation platform.

4.1 Custom installation of the Management Server

As explained in the scenario description, we have three zones in our customers environment, as shown in Figure 4-1.

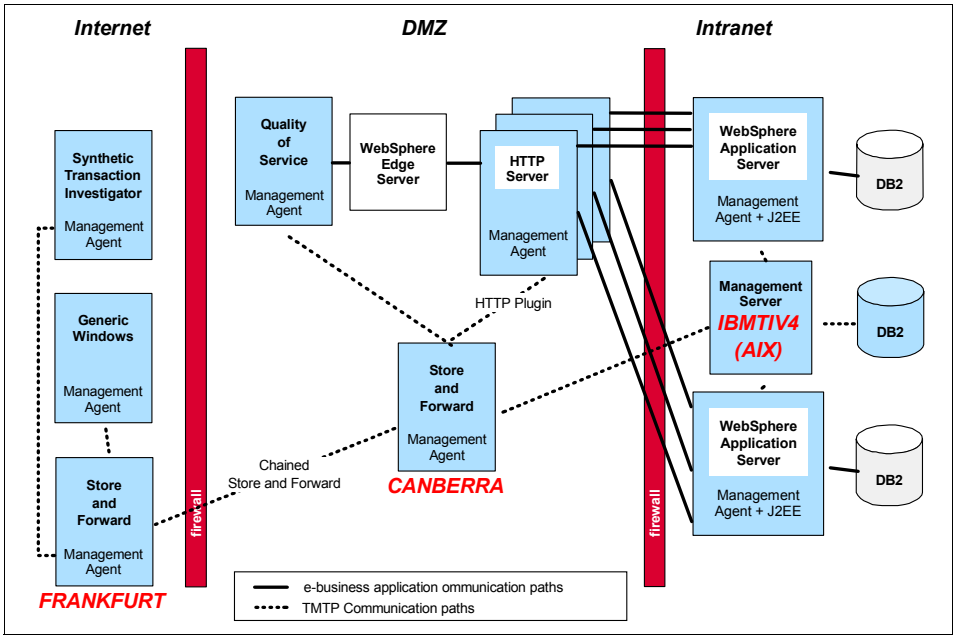


Figure 4-1 Customer production environment

1. The first zone, where the Management Server and the WebSphere Application Servers are, is the intranet zone. The host name of the Management Server is ibmtiv4.
2. The second zone is the DMZ, where the HTTP servers and the WebSphere Edge server are located. In this zone, we will deploy a Store and Forward agent and Management Agents on the rest of the servers. The host name of the Store and Forward agent in this zone is canberra.
3. The last zone is the Internet zone, where we also need to deploy a Store and Forward agent and Management Agents on the client workstations. The host name of the Store and Forward agent in this zone is frankfurt. The canberra Store and Forward agent will be connected directly to the Management Server, while the frankurt Store and Forward agent will be connected directly into the canberra Store and Forward agent. So the Canberra will basically serve as a Management Server for the frankfurt Store and Forward agent.

4.1.1 Management Server custom installation preparation steps

In this section, we will discuss the preparation steps of the Management Server custom installation. We already have installed DB2 Version 8.1 and WebSphere Application Server Version 5.0 with FixPack 1 applied.

Note: The version number of the WebSphere Application Server changes to 5.0.1 from 5.0 after applying WebSphere FixPack 1.

The following steps will be performed:

1. Operating system requirements check
2. File system creation
3. Depot directory creation
4. DB2 configuration
5. WebSphere configuration
6. Port numbers
7. Generating JKS file
8. Generating KDB and STH files
9. Exchanging certificates
10. Environment variables and last checkups

Here are the steps in more detail:

1. Operating system requirements check

   In our scenario, we are using AIX Version 4.3.3 as the host operating system of the Management Server. The required level of this particular version is 4.3.3.10 or higher. We have previously applied the fix pack for this level. To check if the operating system on the correct level, issue the command shown in Example 4-1 (its output is included as well).

Example 4-1 Output of the oslevel -r command

```
# oslevel -r
4330-10
```

2. File system creation

   The installation of the Management Server requires 1.1 GB of free space on AIX: additionally, we also need 1 GB of space for the TMTP database. We have created two file systems, as shown in Table 4-1 on page 89.

Table 4-1 File system creation

File system	Size	Function
/opt/IBM	1.5 GB	The TMTP installation will be performed here.
/opt/IBM/dbtmtp	1 GB	The TMTP database will reside in this directory.
/install	4 GB	This will be the root directory of the installation depot and the temporary installation directory during the product installation. This will be removed once the installation is finished successfully.

3. Depot directory creation

   There are two ways to install the TMTP: either you use the original CDs or you download the installation code. In the second case, you need to create a predefined installation depot directory structure. We are using the second option. The following structure has to be created even if you are using a custom installation scenario; however, you do not have to copy the installation source files into the directories if a product like db2 is already installed.

   a. Create /$installation_root/.

      This will contain the Management Server installation binaries. If you have the packed downloaded version, once you unpack, it will create the following two directories:

      - /$installation_root/lib
      - /$installation_root/keyfiles

      If you are using CDs and you still would like to create a depot, you need to copy the entire content of the CD into the /$installation_root/ directory.

   b. Create /$installation_root/db2.

      This will hold the DB2 installation binaries.

   c. Create /$installation_root/was5.

      This is the location where the WebSphere installation binaries will be copied.

   d. Create /$installation_root/wasFp1

      This is the directory for the WebSphere FixPack 1.

Important: The directory names are case sensitive.

For detailed descriptions of the files and directories to be copied into the specific product directories, please consult the *IBM Tivoli Monitoring for Transaction Performance Installation Guide Version 5.2.0*, SC32-1385.

In our scenario, we have created a file system named /install and use it to serve as the $installation_root. This file system can be removed after the installation.

To provide temporary space for the product installation itself, we have also created the /install/tmp directory.

We have the output shown in Example 4-2 if we execute an **ls -l** command on the /install directory after unpacking the installation files for the Management Server.

Example 4-2 Management Server $installation_root

```
-rwxrwxrwx   1 nuucp    mail         885 Sep 08 09:57 MS.opt
-rwxrwxrwx   1 24       24          1332 Sep 08 09:57 MS_db2_embedded_unix.opt
-rwxrwxrwx   1 23       23           957 Sep 08 09:57 MS_db2_embedded_w32.opt
-rwxrwxrwx   1 13       13         10431 Sep 08 09:57 MsPrereqs.xml
drwxrwsrwx   5 root     sys          512 Sep 12 11:19 db2
-rwxrwxrwx   1 12       12           233 Sep 08 09:57 dm_db2_1.ddl
drwxrwsrwx   2 493      493          512 Sep 19 09:26 keyfiles
drwxrwsrwx   2 493      493          512 Sep 08 09:57 lib
drwxrwxrwx   2 root     system       512 Sep 11 10:08 lost+found
-rwxrwxrwx   1 lpd      printq        12 Sep 08 09:57 media.inf
-rwxrwxrwx   1 11       mqbrkr      3792 Sep 08 09:57 prereqs.dtd
-rwxrwxrwx   1 10       audit      16384 Sep 08 09:57 reboot.exe
-rwxrwxrwx   1 12       12       532041609 Sep 08 09:58 setup_MS.jar
-rwxrwxrwx   1 16       16       18984898 Sep 08 09:58 setup_MS_aix.bin
-rwxrwxrwx   1 15       15            24 Sep 08 09:58 setup_MS_aix.cp
-rwxrwxrwx   1 16       16       20824338 Sep 08 09:58 setup_MS_lin.bin
-rwxrwxrwx   1 15       15            24 Sep 08 09:58 setup_MS_lin.cp
-rwxrwxrwx   1 19       19       19277890 Sep 08 09:58 setup_MS_lin390.bin
-rwxrwxrwx   1 18       18            24 Sep 08 09:58 setup_MS_lin390.cp
-rwxrwxrwx   1 16       16       18960067 Sep 08 09:58 setup_MS_sol.bin
-rwxrwxrwx   1 15       15            24 Sep 08 09:58 setup_MS_sol.cp
-rwxrwxrwx   1 15       15            24 Sep 08 09:58 setup_MS_w32.cp
-rwxrwxrwx   1 16       16       18516023 Sep 08 09:58 setup_MS_w32.exe
-rwxrwxrwx   1 11       mqbrkr      5632 Sep 08 09:58 startpg.exe
drwxrwsrwx   2 root     sys          512 Sep 11 11:21 tmp
-rwxrwxrwx   1 11       mqbrkr     24665 Sep 08 09:58 w32util.dll
drwxrwsrwx   5 root     sys          512 Sep 12 11:12 was5
drwxrwsrwx   7 root     sys          512 Sep 18 18:10 wasFp1
```

4. DB2 configuration

   As we already mentioned, DB2 Version 8.1 is already installed. We need to perform additional steps to enable the setup to run successfully.

   a. As we are emulating a production environment, we have already created a separate db2 instance for the TMTP database. The instance name and user is set to dbtmtp.

      Note: To create a new DB2 instance, you can either use the db2setup program or the **db2icrt** command.

   b. We have to create the TMTP database before we start the installation. You can choose any name for the TMTP database. In this scenario, we name the database TMTP. We perform the following commands in the DB2 text console to create the TMTP database in the previously created /opt/IBM/dbtmtp directory:

      ```
      create database tmtp on /opt/IBM/dbtmtp
      DB20000I  The CREATE DATABASE command completed successfully.
      ```

   c. We also need to create the buffpool32k bufferpool. So we first connect to the database:

      ```
      connect to tmtp

      Database Connection Information

      Database server        = DB2/6000 8.1.0
      SQL authorization ID   = DBTMTP
      Local database alias   = TMTP
      ```

      and create the required bufferpool:

      ```
      create bufferpool buffpool32k size 250 pagesize 32 k
      DB20000I  The SQL command completed successfully
      ```

   d. Now we have finished configuring the DB2.

5. WebSphere configuration

   The most important thing is to make sure that the WebSphere FixPack 1 is applied, because this is a critical prerequisite prior to the installation. To check it out, log on to the WebSphere admin console and click on the **Home** button in the browser window. We see the window shown in Figure 4-2 on page 92.

About your WebSphere Application Server
IBM WebSphere Application Server, 5.0.1
Build Number: ptf1M0312.13
Build Date: 03/29/2003
(c) Copyright IBM Corporation 1996, 2003.

Figure 4-2 WebSphere information screen

Since the version of the WebSphere is 5.0.1, the WebSphere FixPack 1 is applied.

6. Port numbers

   In this scenario we will use the default port numbers for the TMTP installation. These are:

   – Port for non SSL clients: 9081
   – Port for SSL clients: 9446
   – Management Server SSL Console port: 9445
   – Management Server non Secure Console port: 9082

Important: Since we will perform a custom secure installation, the Management Server non Secure Console port is not applicable in this scenario; however, we mention it to show all the possibly required ports. If you wish to perform a nonsecure installation, the Management Server SSL Console port will not be applicable.

The following ports are important for observing the already installed products.

– DB2 8.1:

DB2_dbtmtp	60000/tcp
DB2_dbtmtp_1	60001/tcp
DB2_dbtmtp_2	60002/tcp
DB2_dbtmtp_END	60003/tcp
db2c_dbtmtp	50000/tcp

– WebSphere 5.0.1:

Admin Console port	9090
SOAP connector port	8880

7. Generating JKS files

   In order to secure our environment using Secure Socket Layer (SSL) communication, we have to generate our own JKS files. We will use the WebSphere's ikeyman utility. We need to create three JKS files:

   a. prodms.jks: This will be used by the Management Server.

   b. proddmz.jks: This will be used by the Store and Forward agent and for those Management Agents that will connect to the Management Server through a Store and Forward agent.

   c. prodagent.jks: This will be used by those Management Agents that have direct connections to the Management Server.

   We type the following command to start the ikeyman utility on AIX:

   ```
   ./usr/WebSphere/AppServer/bin/ikeyman.sh
   ```

   This command will take us to the ikeyman dialog shown in Figure 4-3.

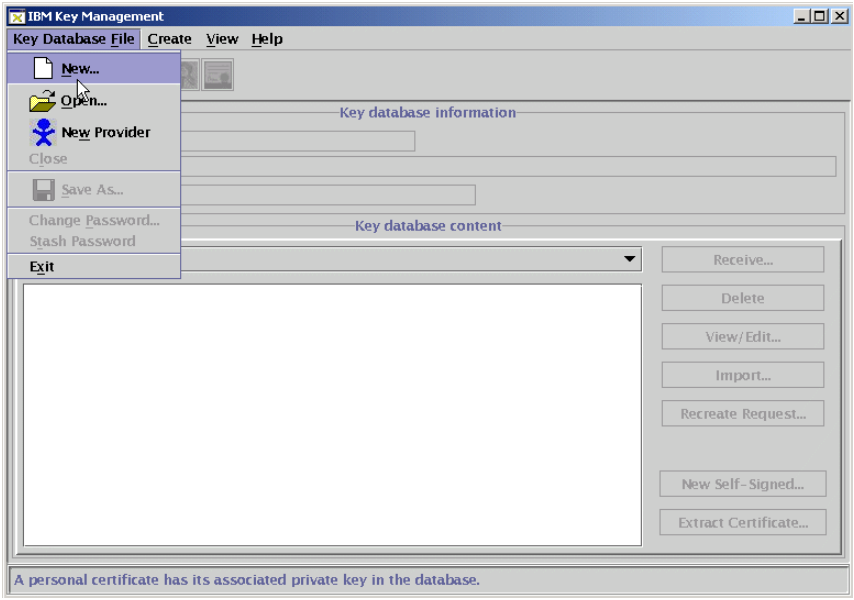


Figure 4-3 ikeyman utility

   – We select the **Key Database File** → **New** option once the ikeyman utility starts.

   – We select **JKS** from the Key Database Type, since this is supported by the TMTP. We name it prodms.jks and set the location to /install/keyfiles to save the file, as shown in Figure 4-4 on page 94.

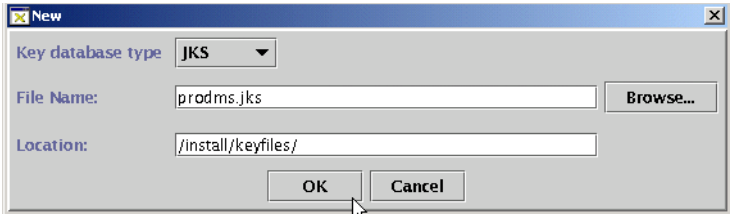


Figure 4-4 Creation of custom JKS file

– At the next screen (Figure 4-5), we provide the password for the JKS file. We have to use this password during the installation of the TMTP product.

Figure 4-5 Set password for the JKS file

– We choose to create a new self signed certificate. We select the **New Self Signed Certificate** from the **Create** menu (see Figure 4-6 on page 95).

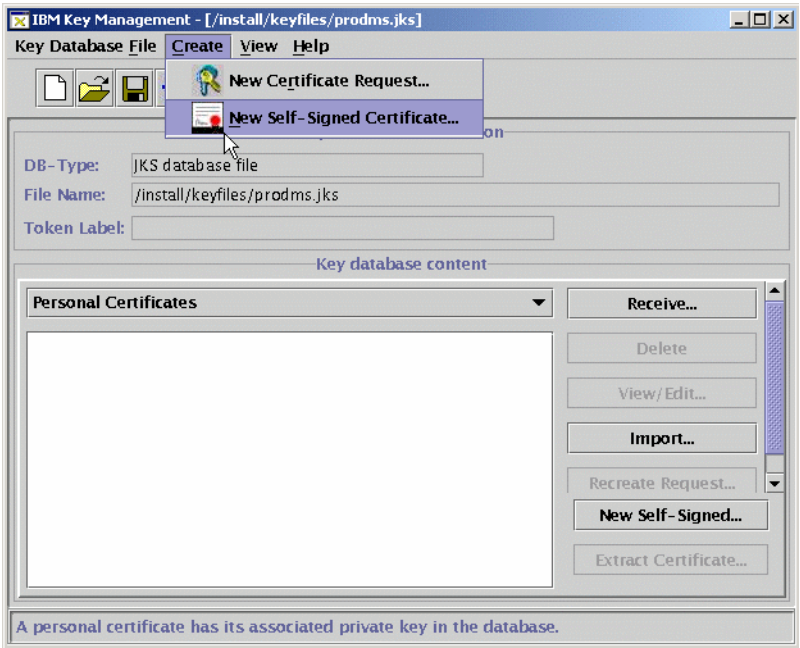


Figure 4-6 Creating a new self signed certificate

Note: At this point, you have the following options: You can purchase a certificate from a Certificate Authority, you can use a pre-existing certificate, or you can create a self signed certificate. We chose the last option.

– In Figure 4-7 on page 96, we define the following:

Key Label	prodms.
Common name	ibmtiv4.itsc.austin.ibm.com, which is the fully qualified host name of the machine where the Management Server will be installed.
Organization	IBM.
Country or Region	US.

We leave the rest of the options on the default setting.

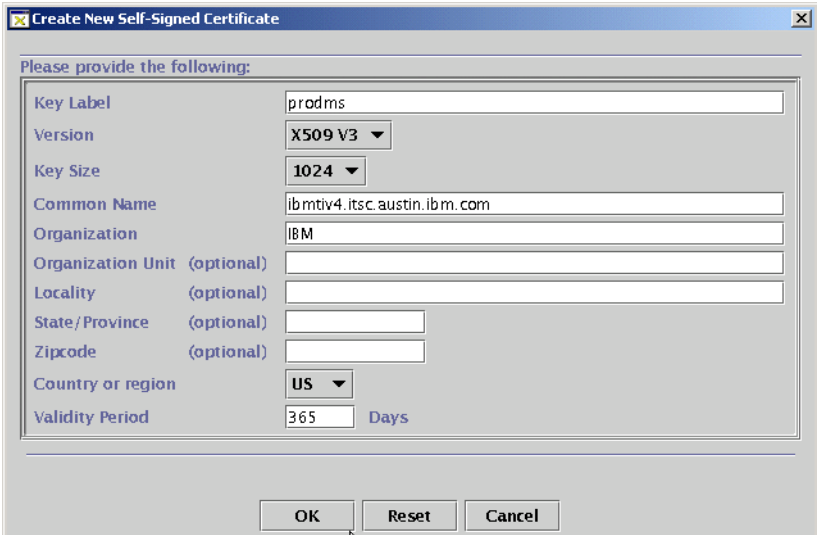


Figure 4-7 New self signed certificate options

– In the next step, shown in Figure 4-8 on page 97, we modify the password of the new self signed certificate by selecting **Key Database File** → **Change Password** and then pressing the **OK** button, as in Figure 4-9 on page 97.

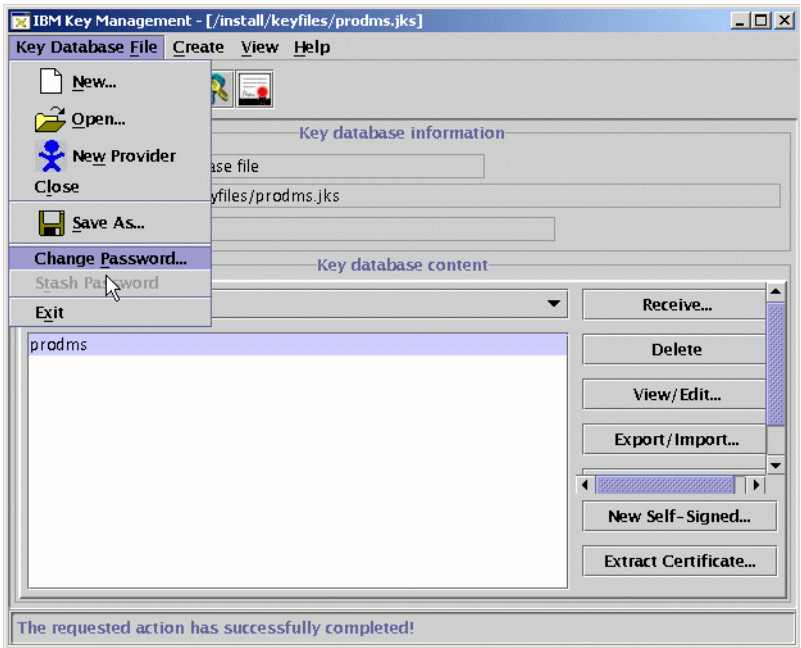


Figure 4-8 Password change of the new self signed certificate

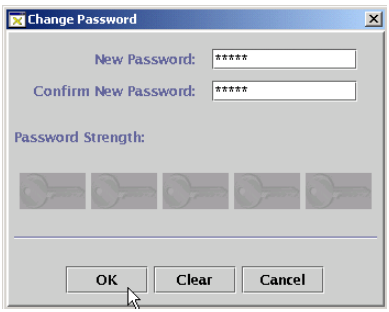


Figure 4-9 Modifying self signed certificate passwords

– Once the password is changed, we are ready to create the JKS file for the Management Server.

  The next step is to create the same JKS files for the Management Agent and for the Store and Forward agent. We use the same steps as above, except for some different parameters, as explained in Table 4-2 on page 98.

Table 4-2 JKS file creation differences

File name	Self signed certificate's name
proddmz.jks	proddmz
prodagent.jks	prodagent

8. Generating KDB and STH files

   Once the JKS files are generated, we need to generate a KDB file and its STH (password) file for the correct secure installation of the WebSphere Caching proxy on the Store and Forward agents. The WebSphere Caching proxy gets installed automatically with the Store and Forward agent. We will generate these files:

   prodsnf.kdb CMS Key Database file

   prodsnf.sth The Password file for the CMS Key Database file

   We have to use a gskit5 tool (provided with the WebSphere Application Server) in installable format. First, we need to install it. The installation files are located under [WebSphereRoot]/gskit5install/, in our case, it is /usr/WebSphere/AppServer/gskit5install/. We execute the installation with the following command:

   ```
   ./gskit.sh
   ```

   The product gets installed to the /usr/opt/ibm/gskkm/ directory. The executable are located in the /usr/opt/ibm/gskkm/bin directory.

   – We start the utility with the following command:

     ```
     ./gsk5ikm
     ```

   – We select the **New** option from the **Key Database File** menu, as in Figure 4-10 on page 99.

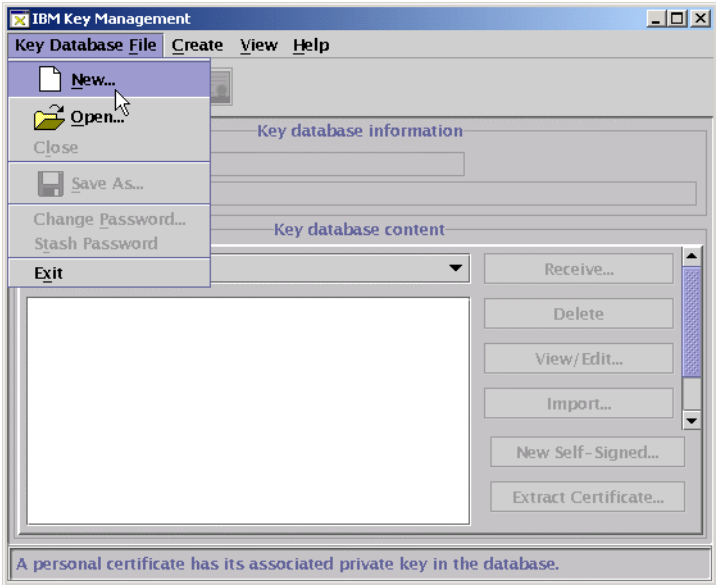


Figure 4-10 GSKit new KDB file creation

– We select the **CMS Key Database file** from the menu. The file name will be prodsnf.kdb (see Figure 4-11).

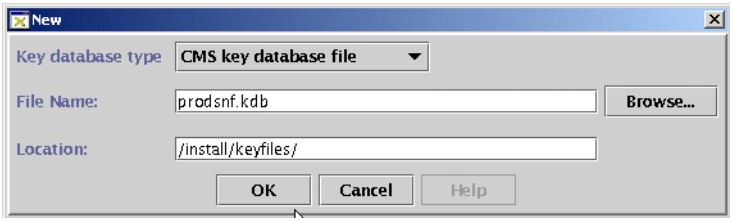


Figure 4-11 CMS key database file creation

– We set the password and select the **Stash the password to a file** option. The stash file name will be prodsnf.sth (see Figure 4-12 on page 100).

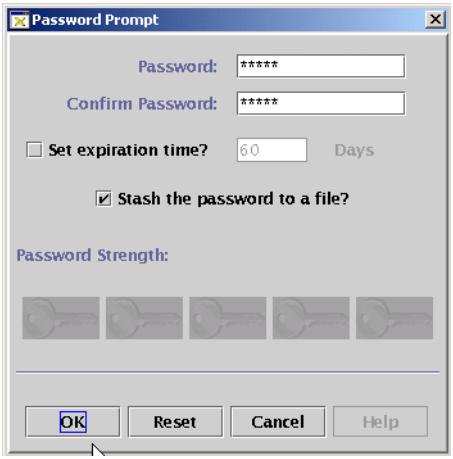


Figure 4-12 Password setup for the prodsnf.kdb

– Now we create a New self signed certificate (see Figure 4-13).

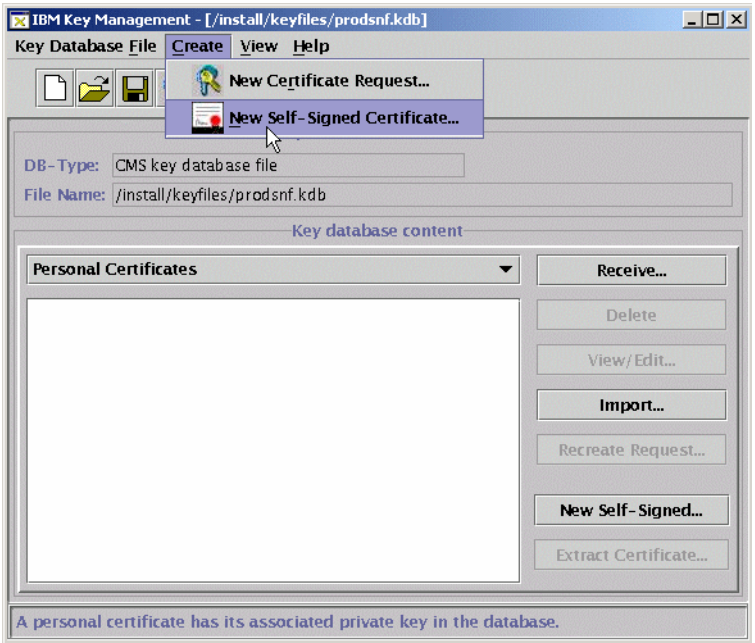


Figure 4-13 New Self Signed Certificate menu

– We name the new certificate prodsnf and the organization IBM. The procedure for the KDB file creation is finished after pressing the **OK** button (see Figure 4-14 on page 101).

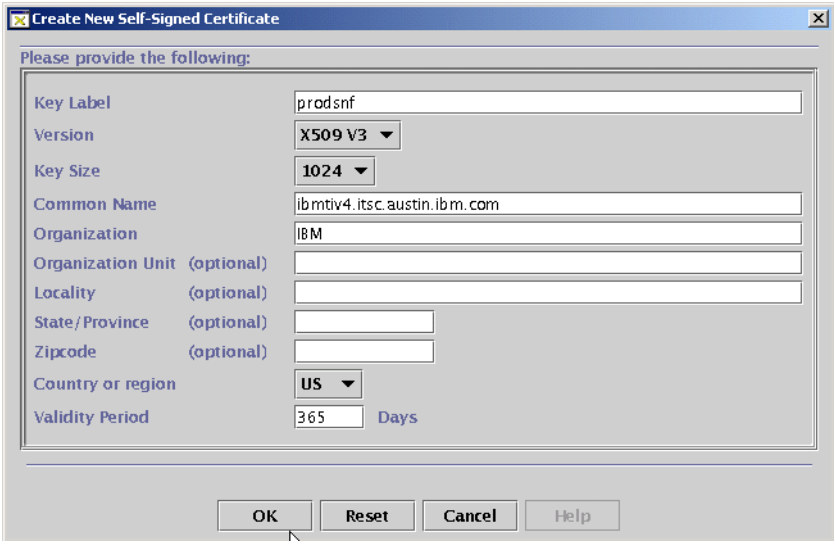


Figure 4-14 Create new self signed certificate

9. Exchanging certificates

   The next step is to exchange the certificates between the JKS and KDB files.

   – In Figure 4-15 on page 102, the.arm files represent the self signed certificates. We have created a self signed certificate for each JKS and KDB file. The next task is to import these certificates into the relevant JKD or KDB files.

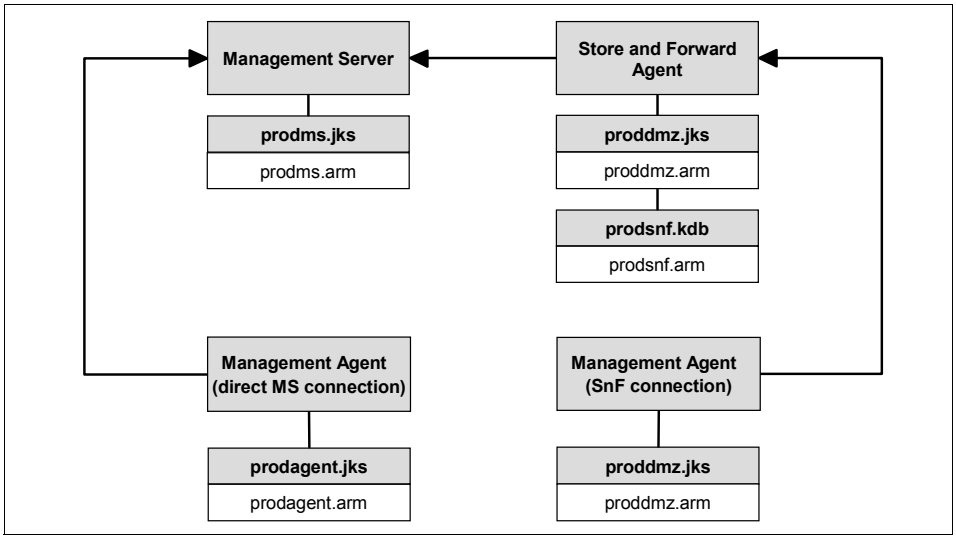


Figure 4-15 Trust files and certificates

– Figure 4-16 on page 103 shows which JKS or KDB file needs to have which self signed certificate:

prodms.jks Needs to have all the certificates.

prodagent.jks Needs to have the certificate from the Management Server and its default certificate. This file will be used for the Management Agents connecting directly to the Management Server.

proddmz.jks Needs to have the certificates from the Management Server and from the prodsnf.kdb file. This file is used for the Store and Forward agent and for its Management Agents in the same zone.

prodsnf.kdb Needs to have the certificate from the Management Server and from the Store and Forward agent's JKS files. This file is used by the WebSphere Caching proxy.

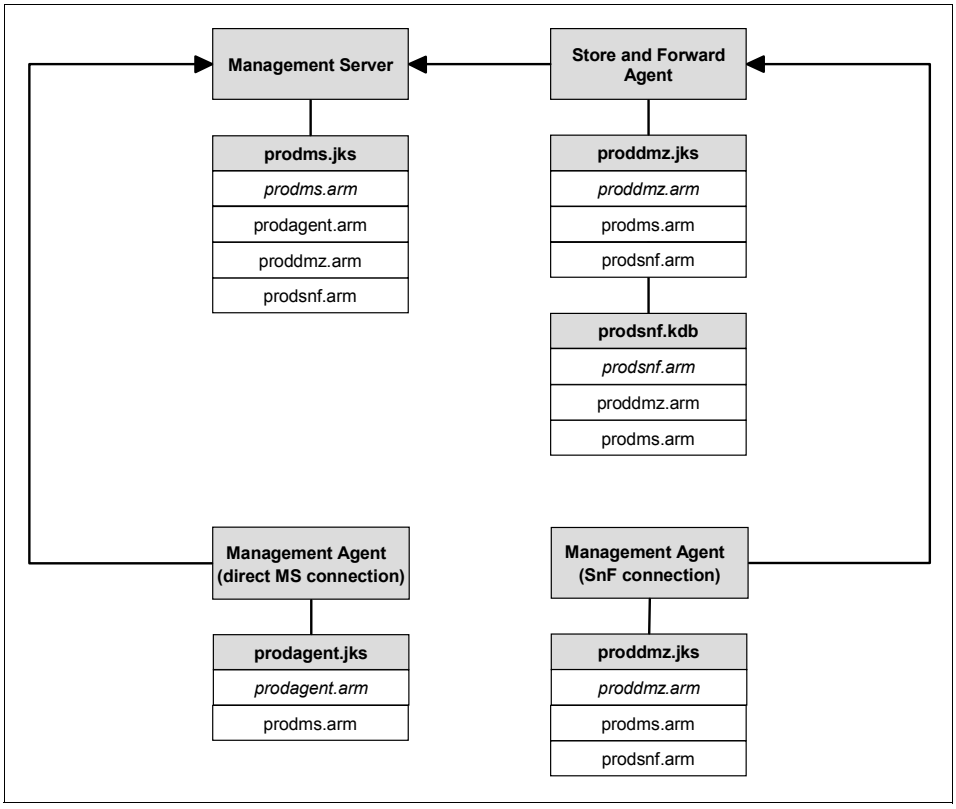


Figure 4-16 The imported certificates

– To exchange the certificates, we have to extract them into .arm files. Start the IBM Key Management tool by executing the following command:

```
./ikeyman.sh
```

– We open the prodms.jks file and press the **Extract Certificate** button (Figure 4-17 on page 104).

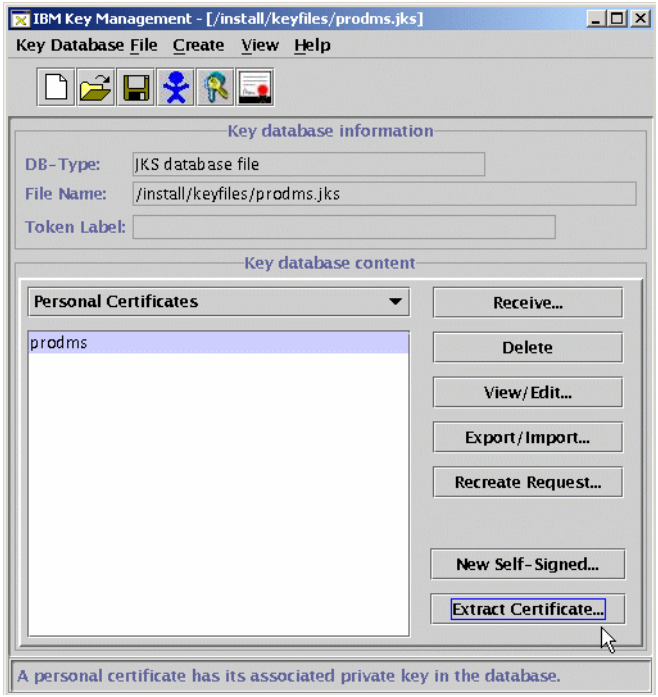


Figure 4-17 Extract Certificate

– We extract the certificate into the **prodms.arm** file (Figure 4-18).

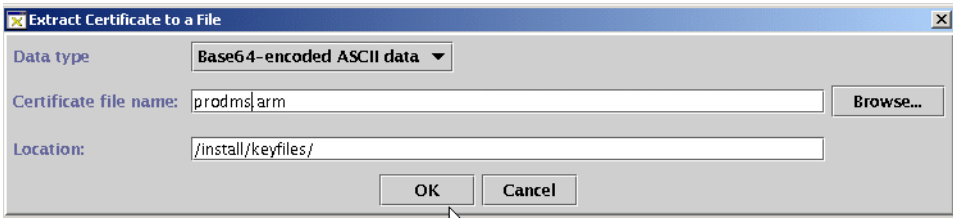


Figure 4-18 Extracting certificate from the msprod.jks file

– Now we add the extracted certificate into the dmzagent.jks file. We open the prodagent.jks file and select the **Signer Certificate** menu from the drop-down menu and press on the **Add** button (Figure 4-19 on page 105).

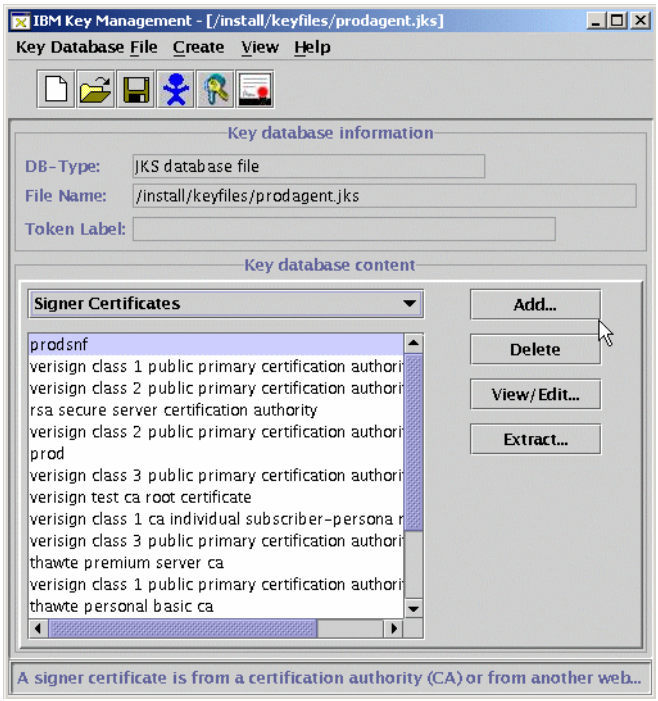

IBM Key Management - [/install/keyfiles/prodagent.jks]
Key Database File Create View Help
Key database information
DB-Type: JKS database file
File Name: /install/keyfiles/prodagent.jks
Token Label:
Key database content
Signer Certificates
Add...
Delete
View/Edit...
Extract...
prodsnf
verisign class 1 public primary certification authori
verisign class 2 public primary certification authori
rsa secure server certification authority
verisign class 2 public primary certification authori
prod
verisign class 3 public primary certification authori
verisign test ca root certificate
verisign class 1 ca individual subscriber-persona r
verisign class 3 public primary certification authori
thawte premium server ca
verisign class 1 public primary certification authori
thawte personal basic ca
A signer certificate is from a certification authority (CA) or from another web...

Figure 4-19 Add a new self signed certificate

– Select the prodms.arm file and press **OK** to add it to the prodagent.jks file (Figure 4-20).

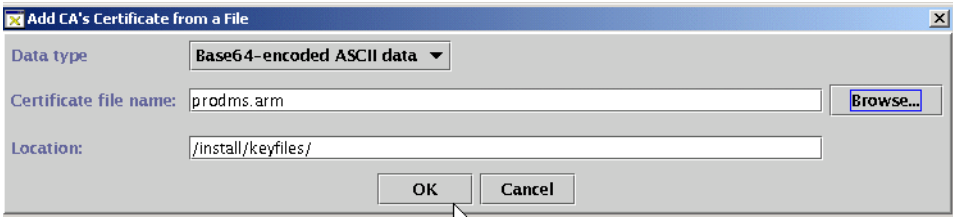


Figure 4-20 Adding a new self signed certificate

– After pressing **OK**, the ikeyman tool asks for the label of the certificate. Use the same name as in the arm file (Figure 4-21 on page 106).

Figure 4-21 Label for the certificate

– The imported certificate is now on the **Signer Certificates** list (Figure 4-22).

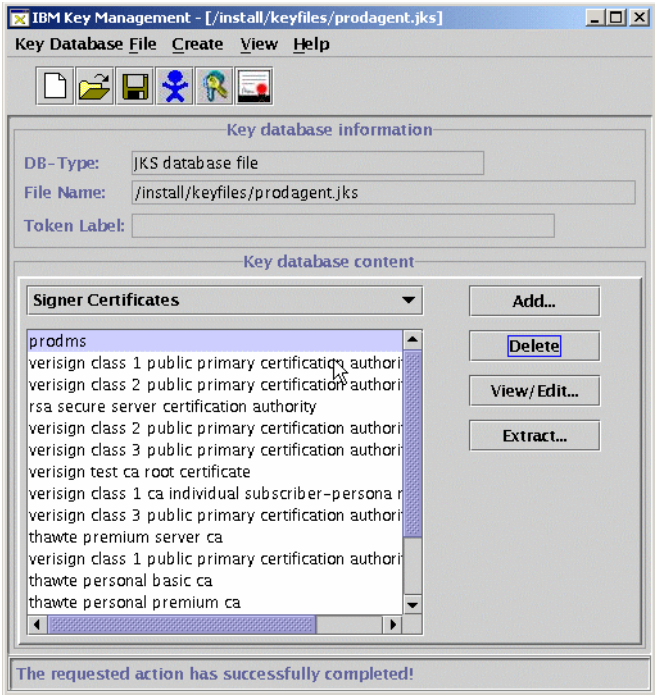

IBM Key Management - [/install/keyfiles/prodagent.jks]
Key Database File Create View Help
Key database information
DB-Type: JKS database file
File Name: /install/keyfiles/prodagent.jks
Token Label:
Key database content
Signer Certificates
Add...
Delete
View/Edit...
Extract...
prodms
verisign class 1 public primary certification authori
verisign class 2 public primary certification authori
rsa secure server certification authority
verisign class 2 public primary certification authori
verisign class 3 public primary certification authori
verisign test ca root certificate
verisign class 1 ca individual subscriber-persona r
verisign class 3 public primary certification authori
thawte premium server ca
verisign class 1 public primary certification authori
thawte personal basic ca
thawte personal premium ca
The requested action has successfully completed!

Figure 4-22 The imported self signed certificate

We follow these steps to extract and add all self signed certificates into the relevant JSK or KDB files.

10. Environment variables

Prior to the installation we have to source the DB2 and WebSphere environment variables as follows:

```
. /usr/WebSphere/AppServer/bin/setupCmdLine.sh
. /home/dbtmtp/sqllib/db2profile
```

This will enable you to set up the program to detect the location and perform actions on DB2 and WebSphere.

Also, set up the variable *$TMPDIR* to define the new temporary installation directory which will be used by the setup program:

```
export TMPDIR=/install/tmp/
```

> **Note:** Before you start the installation, make sure that both the DB2 server and the WebSphere server are up and running.

4.1.2 Step-by-step custom installation of the Management Server

In this section, we will go through the steps of the Management Server installation. As in the previous section, we have prepared our environment for the installation.

- We launch the shell setup program using the following command:

  ```
  ./setup_MS_aix.bin -is:tempdir $TMPDIR
  ```

  The $TMPDIR variable represents the directory where the temporary installation files will be copied.

- Press **Next** in Figure 4-23 on page 108 to proceed to the next window.

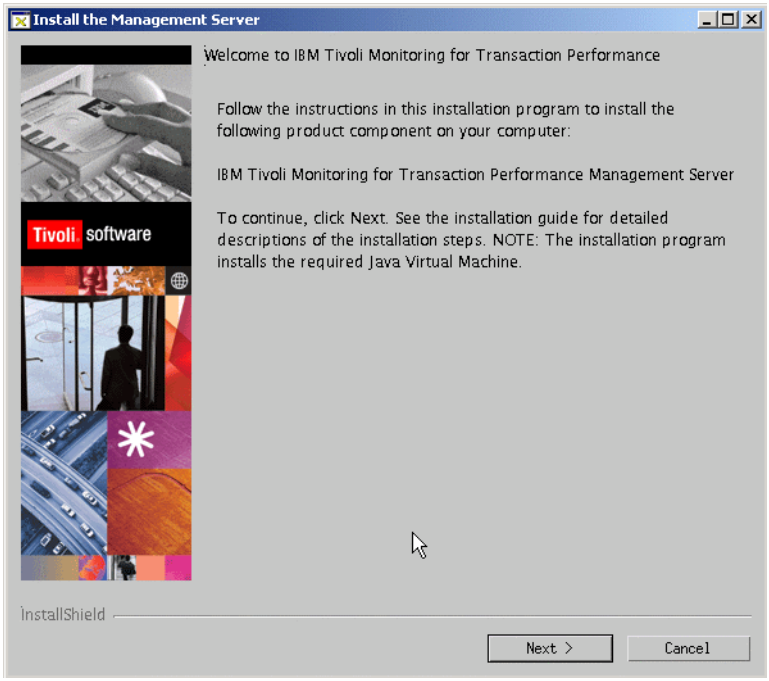

Install the Management Server
Welcome to IBM Tivoli Monitoring for Transaction Performance
Follow the instructions in this installation program to install the following product component on your computer:
IBM Tivoli Monitoring for Transaction Performance Management Server
To continue, click Next. See the installation guide for detailed descriptions of the installation steps. NOTE: The installation program installs the required Java Virtual Machine.
Tivoli software
InstallShield
Next >
Cancel

Figure 4-23 Welcome screen on the Management Server installation wizard

- We accept the license agreement in Figure 4-24 on page 109 and press **Next**.

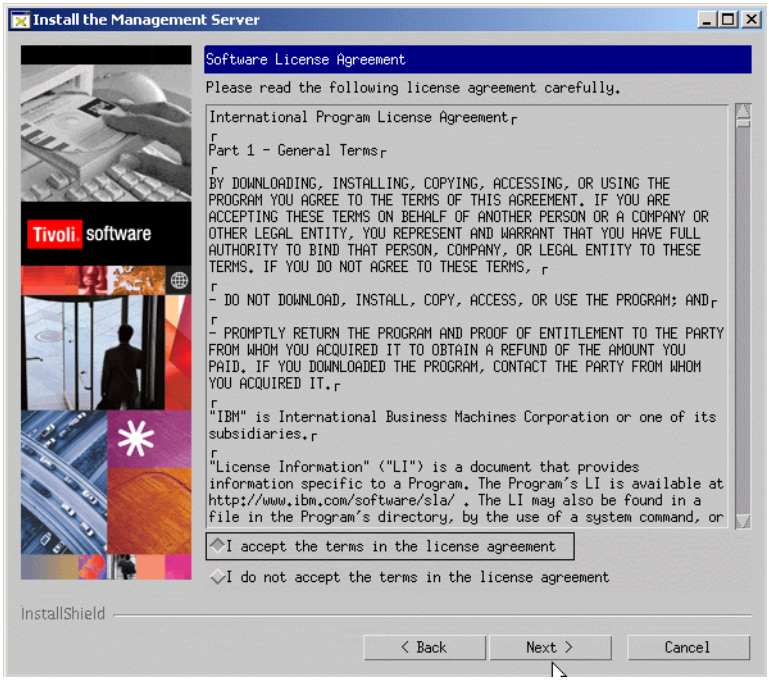

Install the Management Server
Tivoli software
Software License Agreement
Please read the following license agreement carefully.
International Program License Agreement
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND
- PROMPTLY RETURN THE PROGRAM AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM YOU ACQUIRED IT TO OBTAIN A REFUND OF THE AMOUNT YOU PAID. IF YOU DOWNLOADED THE PROGRAM, CONTACT THE PARTY FROM WHOM YOU ACQUIRED IT.
"IBM" is International Business Machines Corporation or one of its subsidiaries.
"License Information" ("LI") is a document that provides information specific to a Program. The Program's LI is available at http://www.ibm.com/software/sla/ . The LI may also be found in a file in the Program's directory, by the use of a system command, or
I accept the terms in the license agreement
I do not accept the terms in the license agreement
InstallShield
< Back
Next >
Cancel

Figure 4-24 License agreement panel

- We leave the installation directory on the default setting (Figure 4-25 on page 110). We have previously created the /opt/IBM file system to serve as installation target.

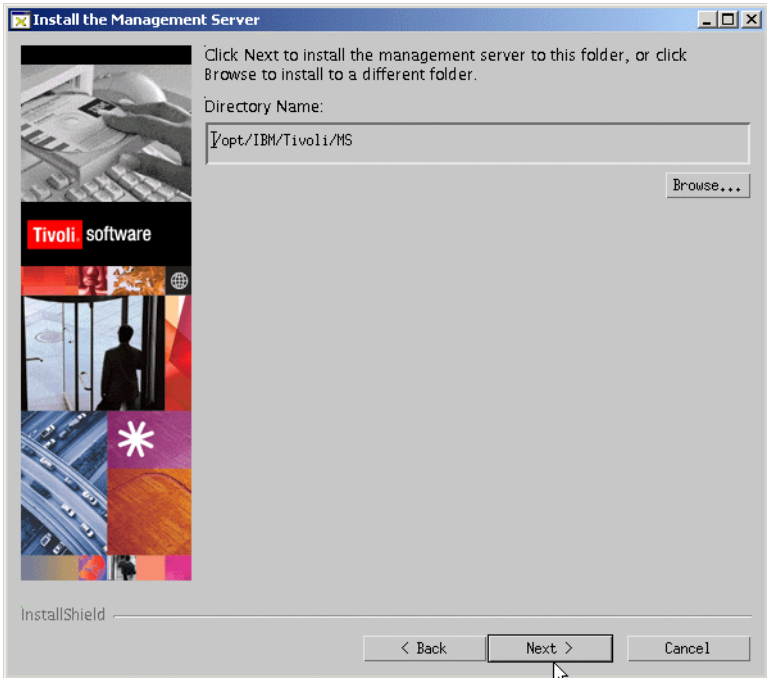


Figure 4-25 Installation target folder selection

- In the next window (Figure 4-26 on page 111), we enable the SSL for Management Server communication. We previously created the prodms.jks file, which serves as the trust and key files. We leave the port settings as the defaults.

Figure 4-26 SSL enablement window

- The installation wizard automatically detects the location of the installed WebSphere if the environment variables are set correctly. In our environment, the WebSphere Application Server security is not enabled, so we unchecked the check box and set the user to root (Figure 4-27 on page 112). Since the WebSphere Application Server security is not enabled, the user you specify here must have root privileges to perform the operation. The installation automatically switches the WebSphere Application Server security *on* once the product was installed and the WebSphere Server has been restarted.

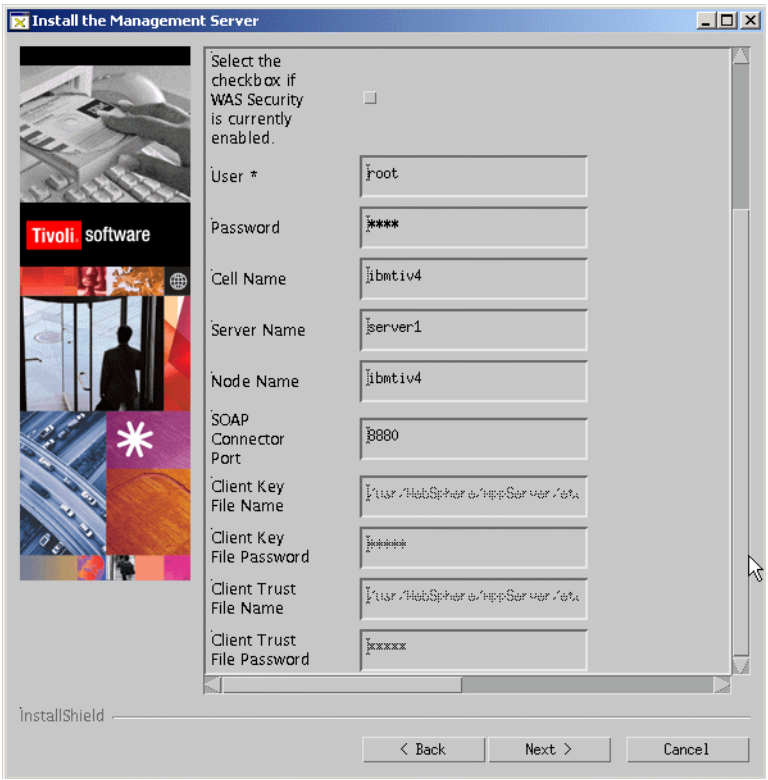


Figure 4-27 WebSphere configuration panel

- As the DB2 database is already installed, we choose for the **Use an existing DB2** database option (Figure 4-28 on page 113).

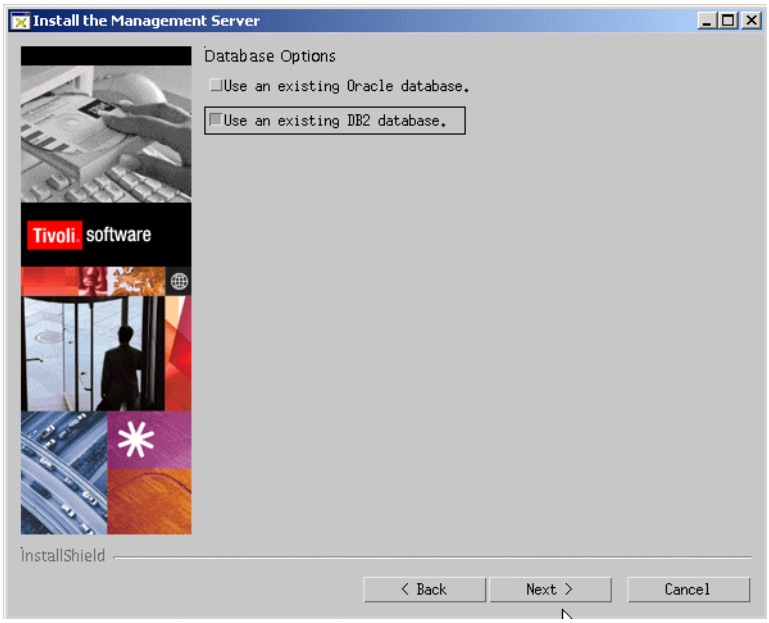


Figure 4-28 Database options panel

- As we already created the dbtmtp db2 instance and the TMTP database on the DB2 level. We choose tmtp for the Database Name, and the database user will be the DB2 instance user dbtmtp. The JDBC path is /home/dbtmtp/sqllib/java/ (see Figure 4-29 on page 114).

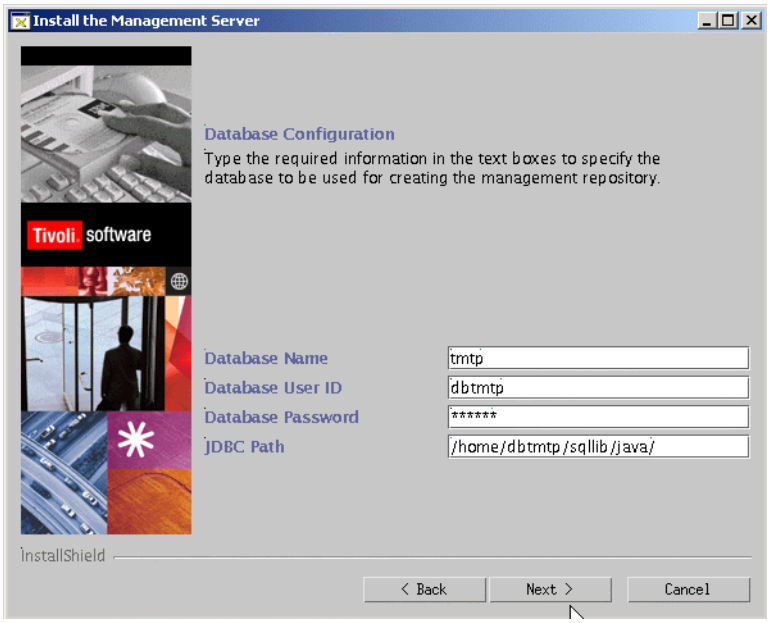


Figure 4-29 Database Configuration panel

Tip: The JDBC path is located under $instance_home/sqllib/java/. So for example, if you use the default instance of the DB2, which is db2inst1, the JDBC path will be /home/db2inst1/sqllib/java/.

- After the DB2 configuration, the setup program reaches the final summarization window (Figure 4-30 on page 115). We press **Next** and the installation of the Management Server starts (Figure 4-31 on page 116).

Figure 4-30 Setting summarization window

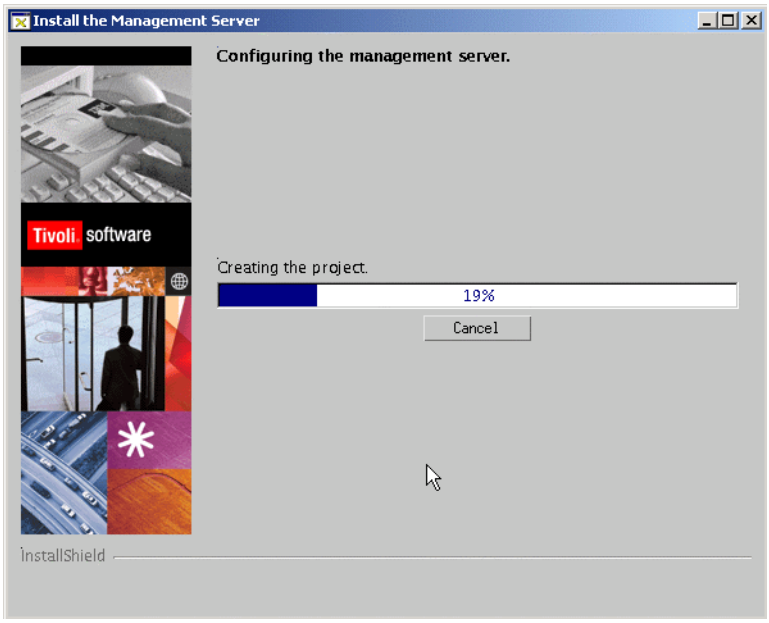


Figure 4-31 Installation progress window

- The installation wizard now creates the TMTP database tables two additional tablespaces: TMTP32K and TEMP_TMTP32K. It also registers the TMTPv5_2 application in the WebSphere Server.
- Once the installation is finished (Figure 4-32 on page 117), the WebSphere Server must be restarted, because the WebSphere Application Server security will now be applied. To stop and start the WebSphere server, we use the following commands. These scripts are located in the *$was_installation_directory*/bin/. In our case, it is /usr/WebSphere/AppServer/bin/.

```
./stopServer.sh server1 -user root -password [password]
./startServer.sh server1 -user root -password [password]
```

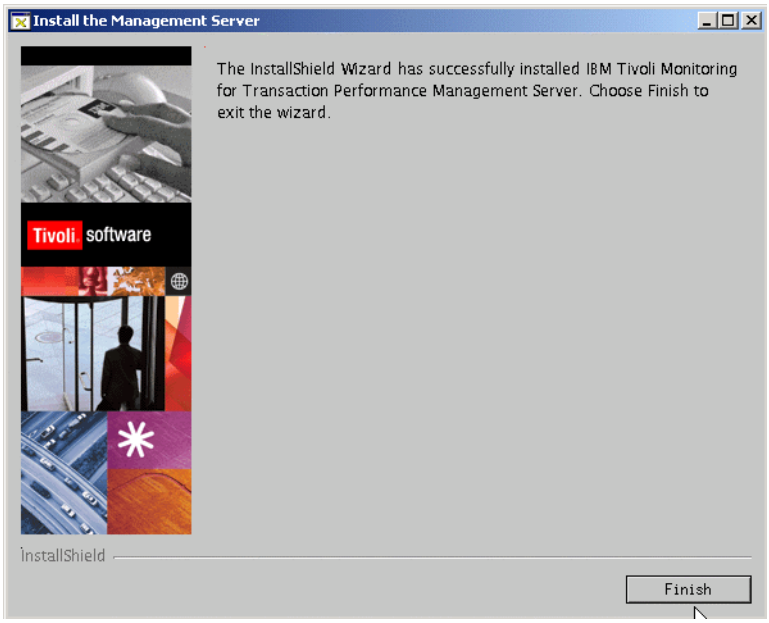

Install the Management Server
The InstallShield Wizard has successfully installed IBM Tivoli Monitoring for Transaction Performance Management Server. Choose Finish to exit the wizard.
Tivoli software
InstallShield
Finish

Figure 4-32 The finished Management Server installation

- Once the WebSphere server is restarted, we log on to the TMTP server by typing the following URL into our browser:

  `https://[ipaddress]:9445/tmtpUI/`

- As the installation was successful, we see the following logon screen in the browser window (Figure 4-33 on page 118).

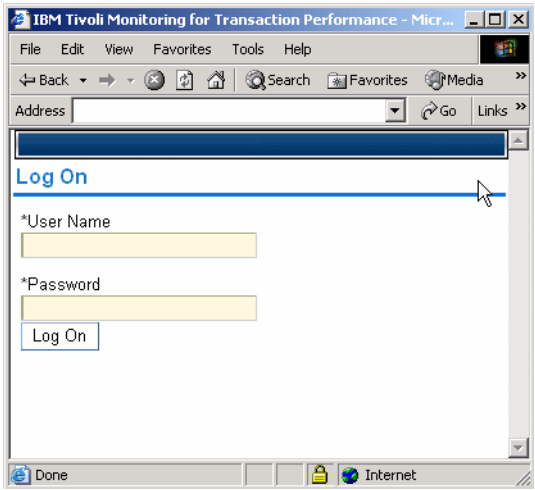


Figure 4-33 TMTP logon window

4.1.3 Deployment of the Store and Forward Agents

In this section, we will deploy the Store and Forward agents into the DMZ and the intranet zone. The following preparations are needed for the installation of the Store and Forward agents:

1. Copy the installation binaries to the local systems. We already did that task. We created the c:\install folder, where we copied the installation binaries for the Store and Forward agent. We copied the binaries of the WebSphere Edge Server Caching proxy to the c:\install\wcp folder.
2. Check to see if the Management Server and Store and Forward agents' fully qualified host names are DNS resolvable.
3. The Store and Forward agents platform will be Windows 2000 Advanced Server with Service Pack 4. The required disk space for all platforms is 50 MB, not including logs.

   The installation wizard will install the following components:

   a. WebSphere Edge Server Caching proxy

   b. Store and Forward agent

   c. We start the installation executing the following command on the Canberra server:

   ```
   setup_SnF_w32.exe -P snfConfig.wcpCdromDir=C:\install\wcp
   ```

   where the -P snfConfig.wcpCdromDir=*directory* specifies the location of the WebSphere Edge Server Caching proxy installation binaries.

Figure 4-34 should appear. Click on **Next**.

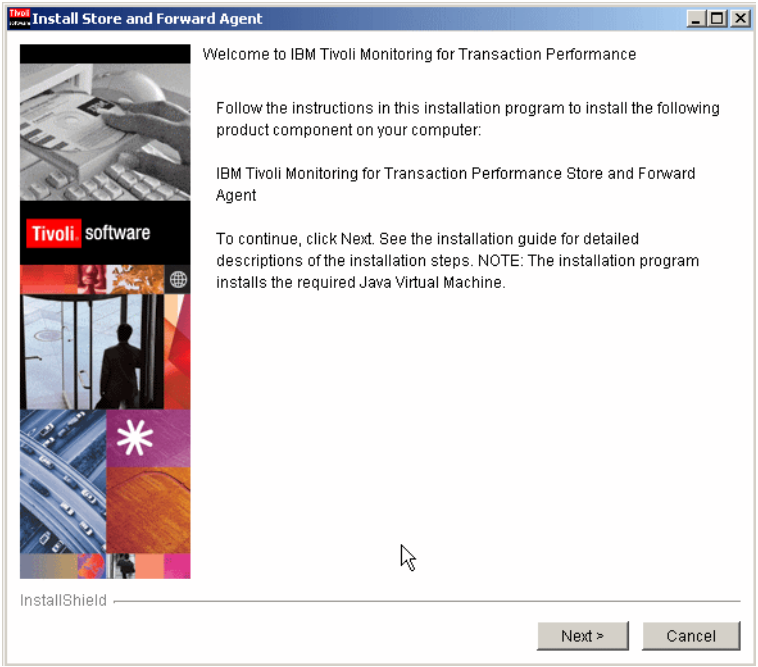

Install Store and Forward Agent
Welcome to IBM Tivoli Monitoring for Transaction Performance
Follow the instructions in this installation program to install the following product component on your computer:
IBM Tivoli Monitoring for Transaction Performance Store and Forward Agent
To continue, click Next. See the installation guide for detailed descriptions of the installation steps. NOTE: The installation program installs the required Java Virtual Machine.
Tivoli software
InstallShield
Next >
Cancel

Figure 4-34 Welcome window of the Store and Forward agent installation

4. In the next window, we accept the License agreement (Figure 4-35 on page 120).

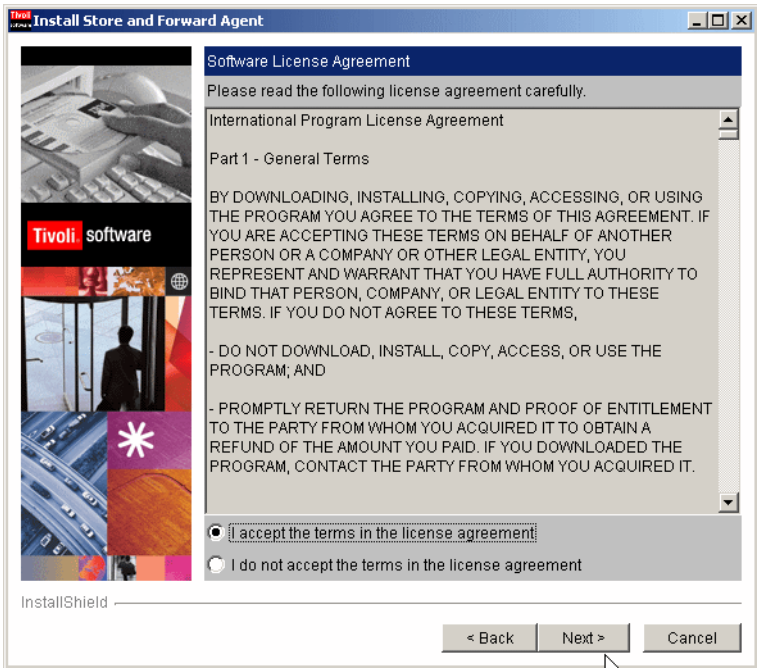

Install Store and Forward Agent
Tivoli software
Software License Agreement
Please read the following license agreement carefully.
International Program License Agreement
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND
- PROMPTLY RETURN THE PROGRAM AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM YOU ACQUIRED IT TO OBTAIN A REFUND OF THE AMOUNT YOU PAID. IF YOU DOWNLOADED THE PROGRAM, CONTACT THE PARTY FROM WHOM YOU ACQUIRED IT.
I accept the terms in the license agreement
I do not accept the terms in the license agreement
InstallShield
< Back
Next >
Cancel

Figure 4-35 License agreement window

- Figure 4-36 on page 121 specifies the installation location of the Store and Forward agent. We leave this on the default setting.

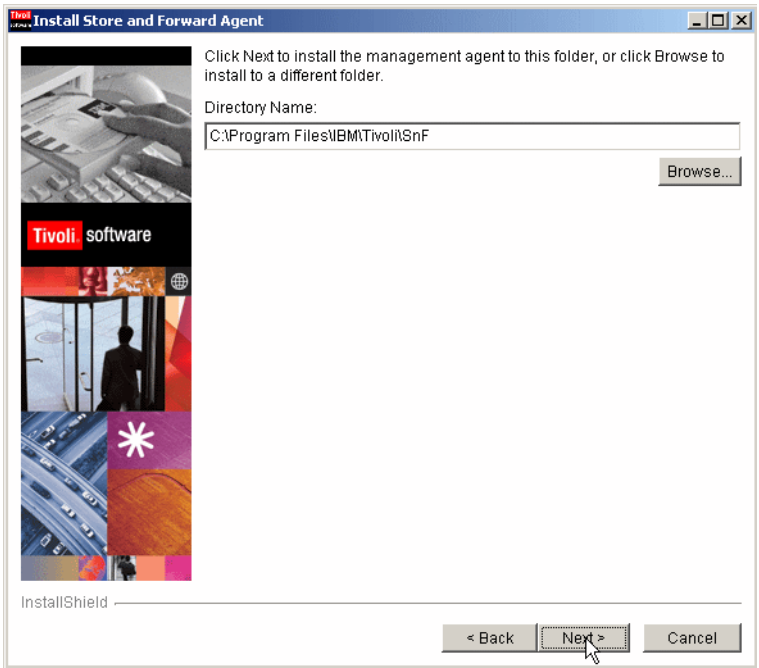

Install Store and Forward Agent
Click Next to install the management agent to this folder, or click Browse to install to a different folder.
Directory Name:
C:\Program Files\IBM\Tivoli\SnF
Browse...
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 4-36 Installation location specification

5. In the first field of Figure 4-37 on page 122, we can specify the Proxy URL. This URL can be either the Management Server itself or in a chained environment and another Store and Forward agent. This specifies the URL where the Store and Forward agent connects to. We specify the Management Server, since this Store and Forward agent is in the DMZ.

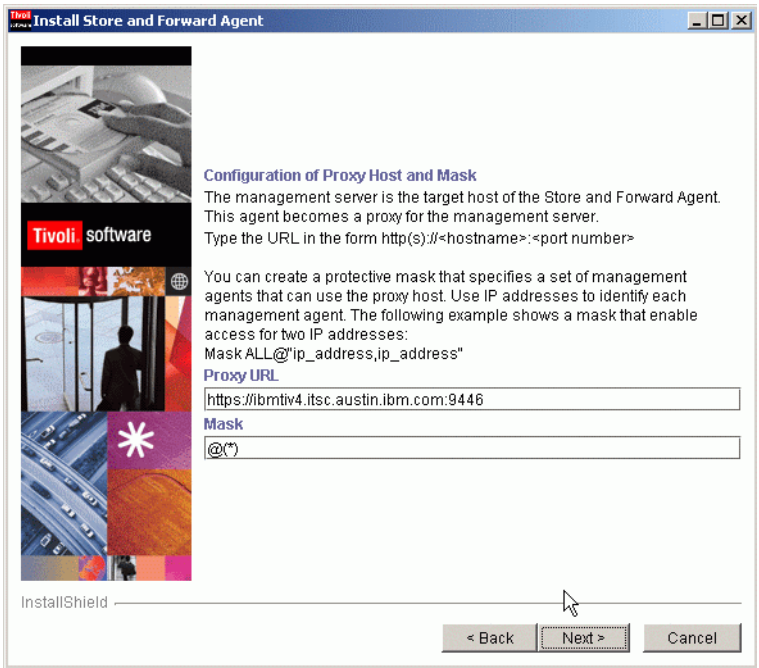

Install Store and Forward Agent
Tivoli software
Configuration of Proxy Host and Mask
The management server is the target host of the Store and Forward Agent. This agent becomes a proxy for the management server.
Type the URL in the form http(s)://<hostname>:<port number>
You can create a protective mask that specifies a set of management agents that can use the proxy host. Use IP addresses to identify each management agent. The following example shows a mask that enable access for two IP addresses:
Mask ALL@"ip_address,ip_address"
Proxy URL
https://ibmtiv4.itsc.austin.ibm.com:9446
Mask
@(*)
InstallShield
< Back
Next >
Cancel

Figure 4-37 Configuration of Proxy host and mask window

As the Management Server has security enabled, we have to specify the protocol as https and the connection port as 9446. The complete URL will be the following:

```
https://ibmtiv4.itsc.austin.ibm.com:9446
```

In the Mask field, we can specify the IP addresses of the computers permitted to access the Management Server through the Store and Forward agent. We choose the @(*) option, which lets all Management Agents connect to this Store and Forward agent in this zone.

6. In Figure 4-38 on page 123, we specify the SSL Key Database and its password stash file. This is required for the installation of the WebSphere Caching proxy. The SSL protocol will be enabled using these files. We are using the custom KEY and STASH files prodsnf.kdb and prodsnf.sth.

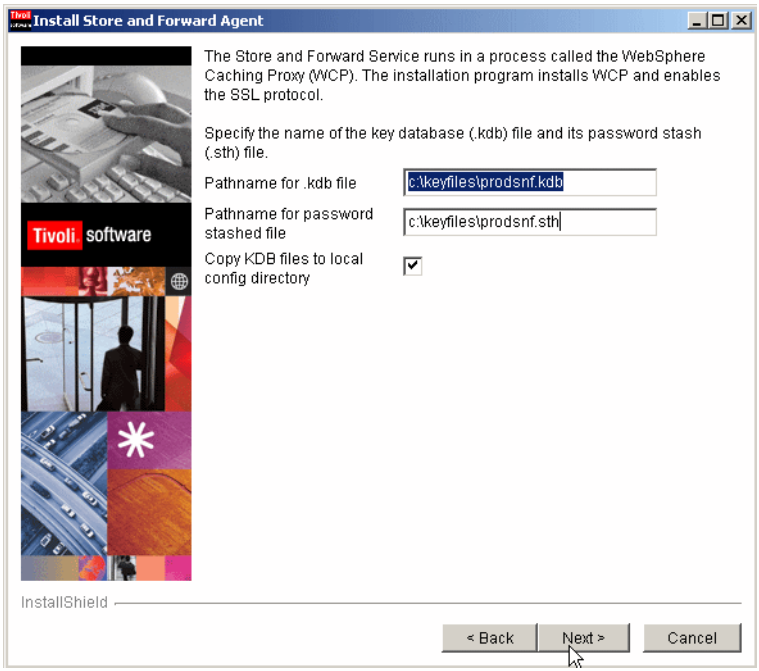

Install Store and Forward Agent
The Store and Forward Service runs in a process called the WebSphere Caching Proxy (WCP). The installation program installs WCP and enables the SSL protocol.
Specify the name of the key database (.kdb) file and its password stash (.sth) file.
Pathname for .kdb file
c:\keyfiles\prodsnf.kdb
Pathname for password stashed file
c:\keyfiles\prodsnf.sth
Copy KDB files to local config directory
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 4-38 KDB file definition

7. In Figure 4-39 on page 124, we have to specify the following things:
   - SnF Host Name: The Store and Forward agent fully qualified host name. In our case, it is canberra.itsc.austin.ibm.com.
   - User Name/User Password: We have to specify a user that has an agent role on the WebSphere Application Server, which is the same as the Management Server in our environment. We specify the root account.
   - Enable SSL: We select this option, since we have a secure installation of the Management Server.
   - We use the Default Port Number, which is 433. This will be the communication port for the Management Agents connecting to this Store and Forward agent.
   - SSL Key store file / SSL Key store file password: We use the previously created JKS file, which is proddmz.jks, and its password.

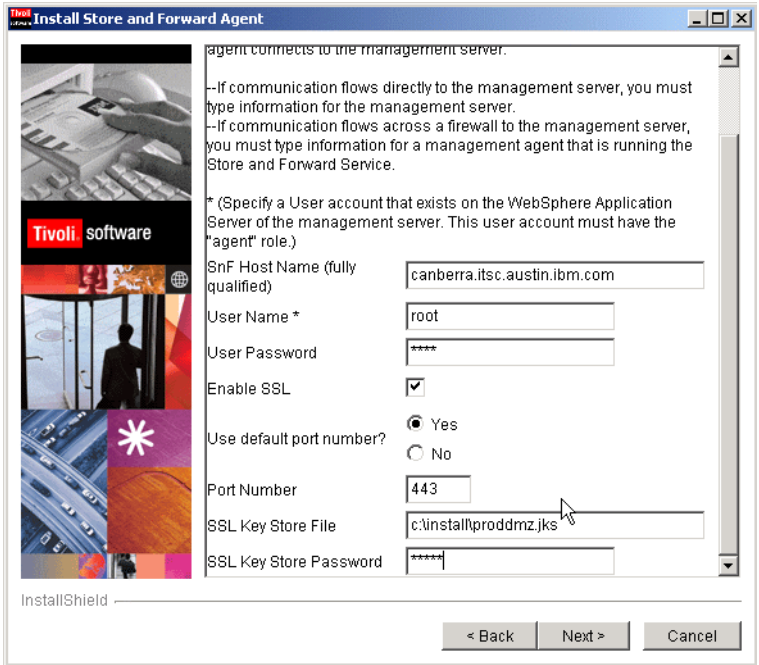

Install Store and Forward Agent
agent connects to the management server.
--If communication flows directly to the management server, you must type information for the management server.
--If communication flows across a firewall to the management server, you must type information for a management agent that is running the Store and Forward Service.
* (Specify a User account that exists on the WebSphere Application Server of the management server. This user account must have the "agent" role.)
SnF Host Name (fully qualified)
canberra.itsc.austin.ibm.com
User Name *
root
User Password
****
Enable SSL
Use default port number?
Yes
No
Port Number
443
SSL Key Store File
c:\install\proddmz.jks
SSL Key Store Password
*****
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 4-39 Communication specification

8. In Figure 4-40 on page 125, we have to specify a local administrative user account what will be used by the Store and Forward agent service. We specify the local Administrator account, which already exists.

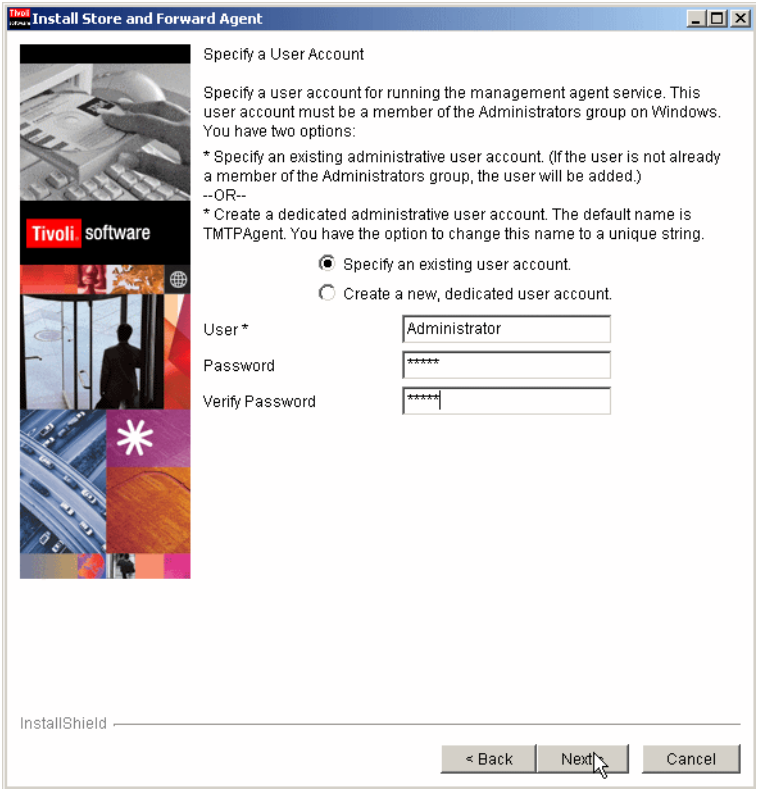

Install Store and Forward Agent
Specify a User Account
Specify a user account for running the management agent service. This user account must be a member of the Administrators group on Windows. You have two options:
* Specify an existing administrative user account. (If the user is not already a member of the Administrators group, the user will be added.)
--OR--
* Create a dedicated administrative user account. The default name is TMTPAgent. You have the option to change this name to a unique string.
Specify an existing user account.
Create a new, dedicated user account.
User *
Administrator
Password
*****
Verify Password
*****
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 4-40 User Account specification window

9. We press **Next** in the window shown in Figure 4-41 on page 126, and the installation starts to install the Store and Forward agent first (Figure 4-42 on page 127).

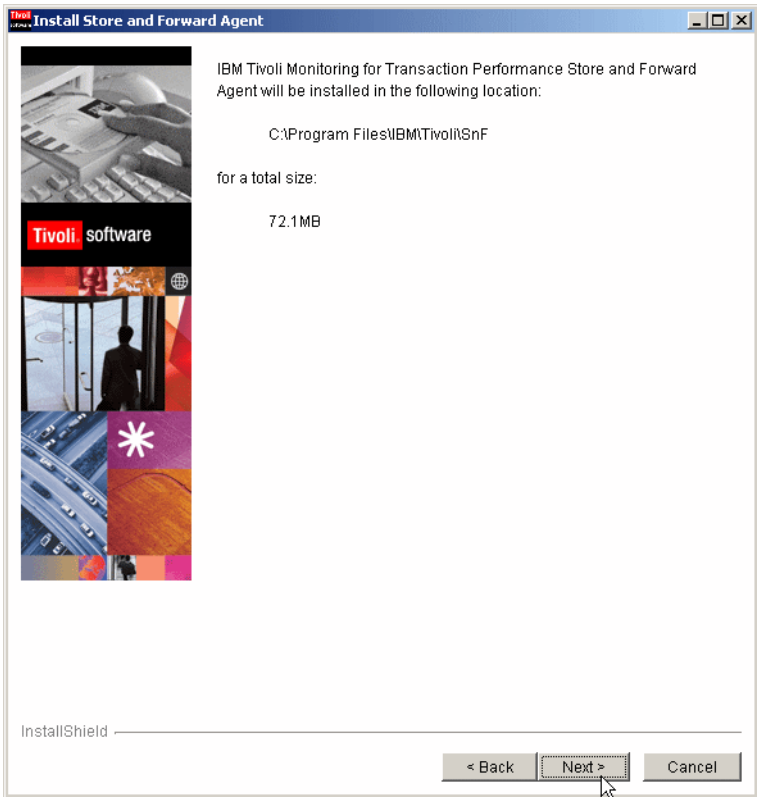

Install Store and Forward Agent
IBM Tivoli Monitoring for Transaction Performance Store and Forward Agent will be installed in the following location:
C:\Program Files\IBM\Tivoli\SnF
for a total size:
72.1 MB
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 4-41 Summary before installation

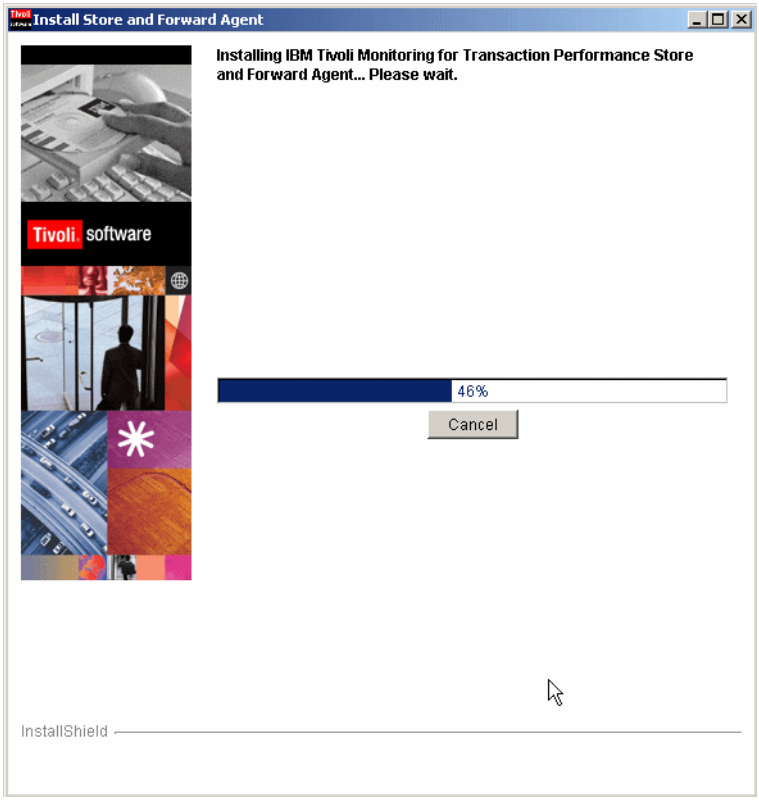


Figure 4-42 Installation progress

10. Once the installation of the Store and Forward agent is completed (Figure 4-43 on page 128), the setup installs the WebSphere Caching proxy. After that, the machine needs to be rebooted. Click on **Next** on the screen shown in Figure 4-43 on page 128.

Install Store and Forward Agent
Installation of the WebSphere Caching Proxy requires a system reboot. The installation program will resume after reboot.
Tivoli software
InstallShield
Next >

Figure 4-43 The WebSphere caching proxy reboot window

11. After the reboot, the installation resumes and configures the WebSphere Caching proxy and the Store and Forward agent. Click on Finish (Figure 4-44 on page 129) to finish the installation.

Install Store and Forward Agent
The InstallShield Wizard has successfully installed IBM Tivoli Monitoring for Transaction Performance Store and Forward Agent. Choose Finish to exit the wizard.
Tivoli software
InstallShield
Finish

Figure 4-44 The final window of the installation

12. We will now deploy the Store and Forward agent for the Internet zone (frankfurt.itsc.austin.ibm.com). This Store and Forward agent will connect to the Store and Forward agent in the DMZ (canberra.itsc.austin.ibm.com). We follow the same installation steps for the previous Store and Forward agent. The different parameters can be found Table 4-3.

Table 4-3 Internet Zone SnF different parameters

Parameter	Value
Proxy URL	`https://canberra.itsc.austin.ibm.com:443`
SnF Host Name (fully qualified)	frankfurt.itsc.austin.ibm.com

Note: The User Name/user password fields are still referring to the *root* user on the Management Server, since this user ID needs to have access to the WebSphere Application Server.

4.1.4 Installation of the Management Agents

We will cover the installation of the Management Agents in this section. As we have the mentioned, we have three zones, and each Management Agent will log on to the Management Server using its zone's Store and Forward agent, or, if the Management Agent is located in the intranet zone, it will log on directly to the Management Server. We first install the Management Agent for the intranet zone. The following pre-checks are required:

1. Check if the Management Server and Store and Forward agents' fully qualified host names are DNS resolvable.
2. The Management Agent's platform will be Windows 2000 Advanced Server with Service Pack 4. The required disk space for all platforms is 50 MB, not including logs.
3. The installation wizard will install the following components:
   - Management Agent
4. We start the installation wizard by executing the following program:

   `setup_MA_w32.exe`

   You should get the window shown in Figure 4-45.

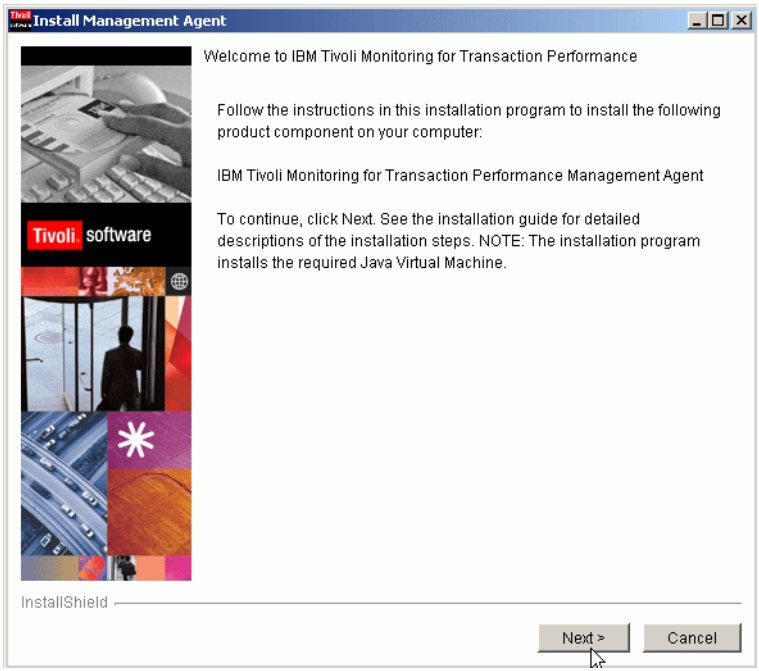

Install Management Agent
Welcome to IBM Tivoli Monitoring for Transaction Performance
Follow the instructions in this installation program to install the following product component on your computer:
IBM Tivoli Monitoring for Transaction Performance Management Agent
To continue, click Next. See the installation guide for detailed descriptions of the installation steps. NOTE: The installation program installs the required Java Virtual Machine.
Tivoli software
InstallShield
Next >
Cancel

Figure 4-45 Management Agent installation welcome window

5. We accept the license agreement and click on the **Next** button (Figure 4-46).

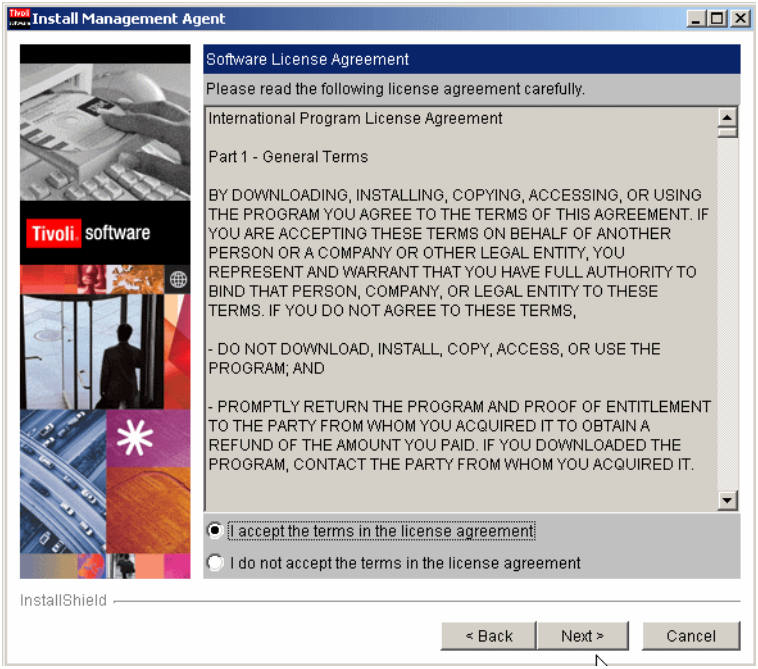

Install Management Agent
Tivoli software
Software License Agreement
Please read the following license agreement carefully.
International Program License Agreement
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND
- PROMPTLY RETURN THE PROGRAM AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM YOU ACQUIRED IT TO OBTAIN A REFUND OF THE AMOUNT YOU PAID. IF YOU DOWNLOADED THE PROGRAM, CONTACT THE PARTY FROM WHOM YOU ACQUIRED IT.
I accept the terms in the license agreement
I do not accept the terms in the license agreement
InstallShield
< Back
Next >
Cancel

Figure 4-46 License agreement window

- We leave the default location for the Management Agent target directory. Click **Next** (Figure 4-47 on page 132).

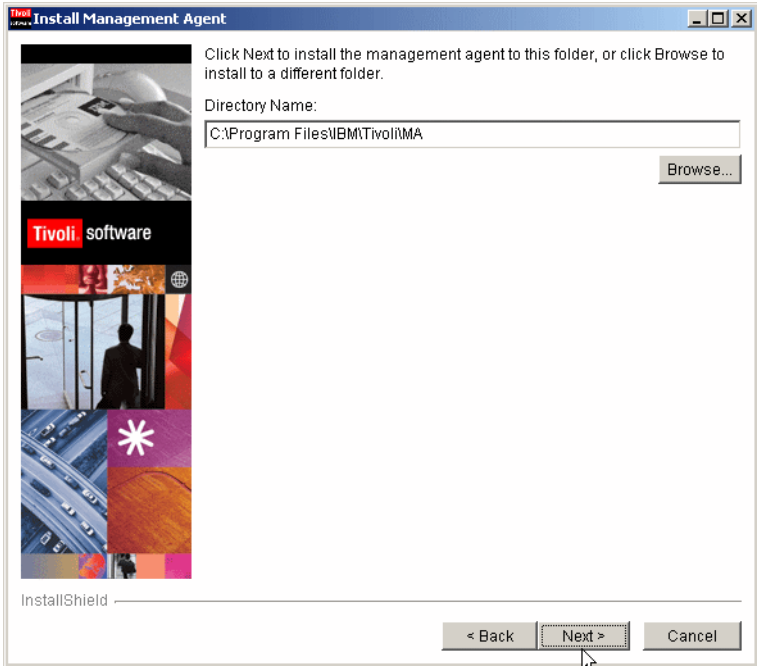


Figure 4-47 Installation location definition

6. In Figure 4-48 on page 133, we specify the parameters for the Management Agent connection.
   - Host Name: As we are in the intranet zone, the Management Agent will directly connect to the Management Server. We specify the Management Server's host name as ibmtiv4.itsc.austin.ibm.com.
   - User Name / User Password: We have to specify a user that has the agent role on the WebSphere Application Server, which is the same as the Management Server in our environment. We specify the root account.
   - Enable SSL: We select this option, since we have a secure installation of the Management Server.
   - Use default port number: As the Management Server is using the default port number, we select **Yes** at this option.
   - Proxy protocol/Proxy Host/Port number: As we are not using proxy, we specify the **No proxy** option.
   - SSL Key Store file/password: We previously created a custom JKS file to serve the agent connections, so we specify the prodagent.jks file and its password.

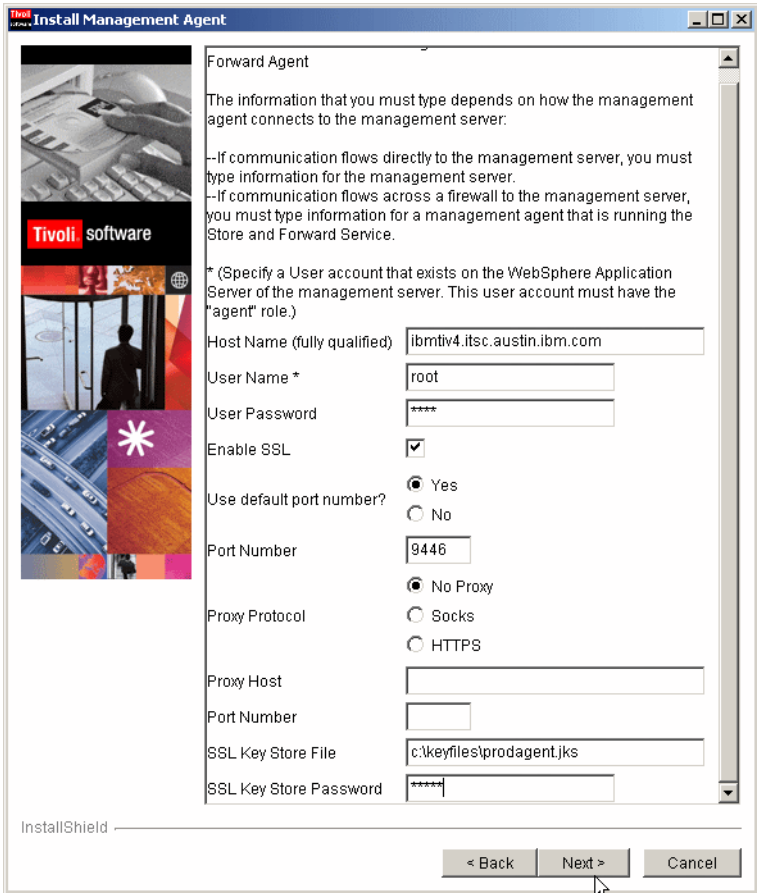

Install Management Agent
Tivoli software
Forward Agent
The information that you must type depends on how the management agent connects to the management server:
--If communication flows directly to the management server, you must type information for the management server.
--If communication flows across a firewall to the management server, you must type information for a management agent that is running the Store and Forward Service.
* (Specify a User account that exists on the WebSphere Application Server of the management server. This user account must have the "agent" role.)
Host Name (fully qualified)
ibmtiv4.itsc.austin.ibm.com
User Name *
root
User Password
****
Enable SSL
Use default port number?
Yes
No
Port Number
9446
Proxy Protocol
No Proxy
Socks
HTTPS
Proxy Host
Port Number
SSL Key Store File
c:\keyfiles\prodagent.jks
SSL Key Store Password
*****
InstallShield
< Back
Next >
Cancel

Figure 4-48 Management Agent connection window

7. In Figure 4-49 on page 134, we specify a local administrative user account that will be used by the Management Agent service. We specify the local Administrator account, which already exists.

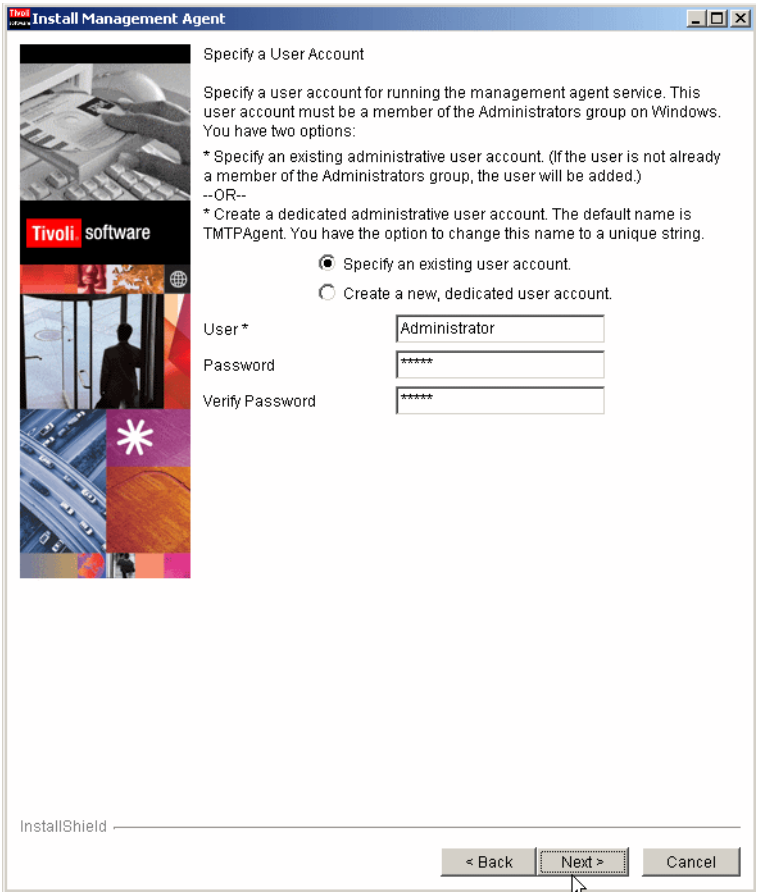

Install Management Agent
Specify a User Account
Specify a user account for running the management agent service. This user account must be a member of the Administrators group on Windows. You have two options:
* Specify an existing administrative user account. (If the user is not already a member of the Administrators group, the user will be added.)
--OR--
* Create a dedicated administrative user account. The default name is TMTPAgent. You have the option to change this name to a unique string.
Tivoli software
Specify an existing user account.
Create a new, dedicated user account.
User *
Administrator
Password
*****
Verify Password
*****
InstallShield
< Back
Next >
Cancel

Figure 4-49 Local user account specification

8. We press **Next** on the installation summary window (Figure 4-50 on page 135).

Figure 4-50 Installation summary window

Press the **Finish** button in the window shown in Figure 4-51 on page 136 to finish the installation.

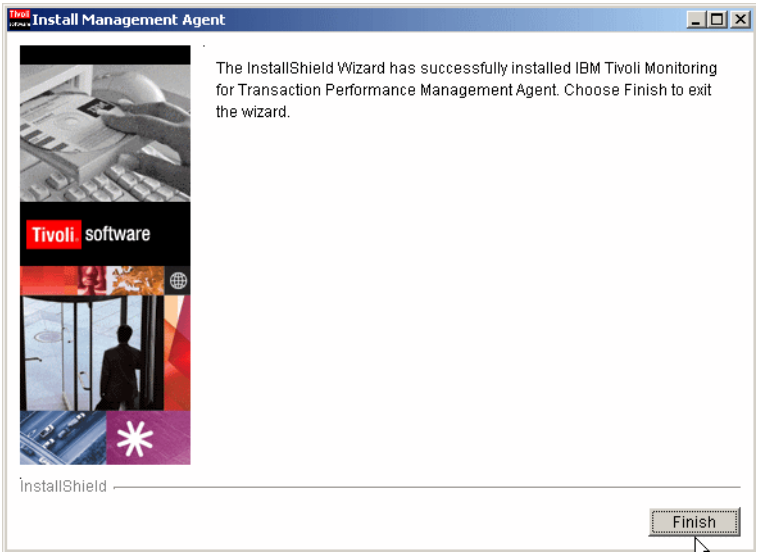

Install Management Agent
The InstallShield Wizard has successfully installed IBM Tivoli Monitoring for Transaction Performance Management Agent. Choose Finish to exit the wizard.
Tivoli software
InstallShield
Finish

Figure 4-51 The finished installation

9. All Management Agents must be installed with the same parameters in the intranet zone. Table 4-4 summarizes the changed parameters for the Management Agent installation in the DMZ and the Internet zone.

Table 4-4 Changed option of the Management Agent installation/zone

Parameter	DMZ	Internet zone
Host Name (The host name of the Store and Forward agent in the specified zone)	Canberra	Frankfurt
Port Number (The default port number of the Store and Forward agent)	443	443
SSL Key Store File/password	dmzagent.jks	dmzagent.jks

Note: The User Name/user password fields are still referring to the *root* user on the Management Server, since this user ID needs to have access to the WebSphere Application Server.

4.2 Typical installation of the Management Server

In this section, we will demonstrate the typical nonsecure installation of the Management Server on SuSE Linux Version 7.3. There are no additional operating system patches needed.

We will use the root file system to perform the installation. On this file system, we have 6 GB of free space, which will be enough for the TMTP installation. The installation wizard will install the following software for us:

- DB2 Server Version 8.1 UDB
- WebSphere Application Server Version 5.0
- WebSphere Application Server Version 5.0 with FixPack 1
- TMTP Version 5.2 Management Server

The DB2 and the WebSphere installation binaries come with the TMTP installation CDs. In order to perform a smooth installation, we created the installation depot, as described in 4.1.2, “Step-by-step custom installation of the Management Server” on page 107, and copied all the necessary products to the relevant directories. Our installation depot location is /install. The output of the **`ls -l /install`** is shown in Example 4-3.

Example 4-3 View install depot

```
tmtp-linux:/sbin # ls -l /install
total 1233316
drwxr-xr-x    7 root     root         4096 Sep 16 08:26 .
drwxr-xr-x   20 root     root         4096 Sep 16 12:06 ..
-rw-r--r--    1 root     root          885 Sep  8 09:57 MS.opt
-rw-r--r--    1 root     root         1332 Sep  8 09:57
MS_db2_embedded_unix.opt
-rw-r--r--    1 root     root          957 Sep  8 09:57 MS_db2_embedded_w32.opt
-rw-r--r--    1 root     root        10431 Sep  8 09:57 MsPrereqs.xml
drwxr-xr-x    5 root     root         4096 Sep 16 04:53 db2
-rw-r--r--    1 root     root          233 Sep  8 09:57 dm_db2_1.ddl
drwxr-xr-x    2 root     root         4096 Sep  8 09:57 keyfiles
drwxr-xr-x    4 root     root         4096 Sep 18 15:49 lib
-rw-r--r--    1 root     root           12 Sep  8 09:57 media.inf
-rw-r--r--    1 root     root         3792 Sep  8 09:57 prereqs.dtd
-rw-r--r--    1 root     root        16384 Sep  8 09:57 reboot.exe
-rw-r--r--    1 root     root     532041609 Sep  8 09:58 setup_MS.jar
-rw-r--r--    1 root     root     18984898 Sep  8 09:58 setup_MS_aix.bin
-rw-r--r--    1 root     root           24 Sep  8 09:58 setup_MS_aix.cp
-rwxr-xr-x    1 root     root     20824338 Sep  8 09:58 setup_MS_lin.bin
-rw-r--r--    1 root     root           24 Sep  8 09:58 setup_MS_lin.cp
-rw-r--r--    1 root     root     19277890 Sep  8 09:58 setup_MS_lin390.bin
-rw-r--r--    1 root     root           24 Sep  8 09:58 setup_MS_lin390.cp
```

```
-rw-r--r--    1 root     root     18960067 Sep  8 09:58 setup_MS_sol.bin
-rw-r--r--    1 root     root           24 Sep  8 09:58 setup_MS_sol.cp
-rw-r--r--    1 root     root           24 Sep  8 09:58 setup_MS_w32.cp
-rw-r--r--    1 root     root     18516023 Sep  8 09:58 setup_MS_w32.exe
-rw-r--r--    1 root     root         5632 Sep  8 09:58 startpg.exe
-rw-r--r--    1 root     root        24665 Sep  8 09:58 w32util.dll
drwxr-xr-x    5 root     root         4096 Sep 16 04:54 was5
drwxr-xr-x    7 root     root         4096 Sep 16 09:32 wasFp1
```

- We start the installation by executing the following command:

  ```
  ./setup_MS_lin.bin
  ```

- At the management Server installation welcome screen, we press **Next** (Figure 4-52).

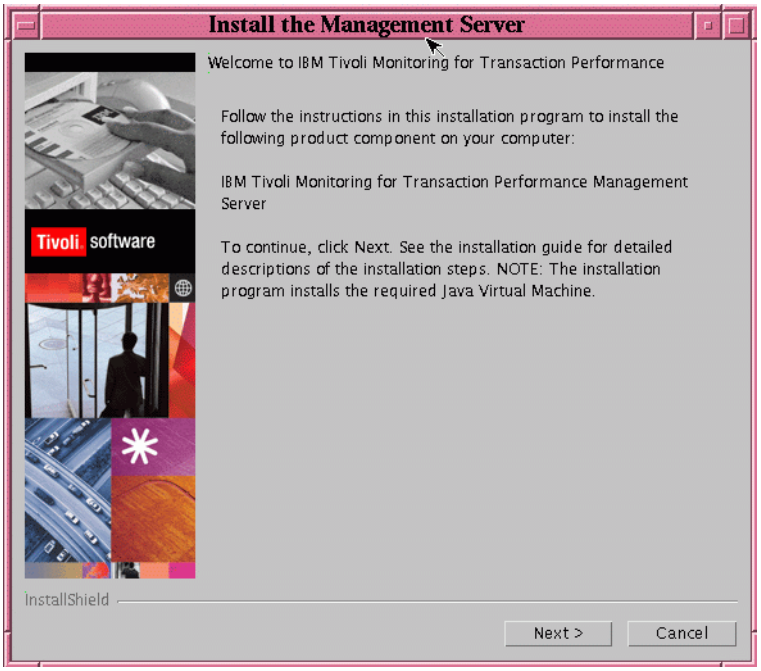

Install the Management Server
Welcome to IBM Tivoli Monitoring for Transaction Performance
Follow the instructions in this installation program to install the following product component on your computer:
IBM Tivoli Monitoring for Transaction Performance Management Server
To continue, click Next. See the installation guide for detailed descriptions of the installation steps. NOTE: The installation program installs the required Java Virtual Machine.
Tivoli software
InstallShield
Next >
Cancel

Figure 4-52 Management Server Welcome screen

- We accept the license agreement and press **Next** (Figure 4-53 on page 139).

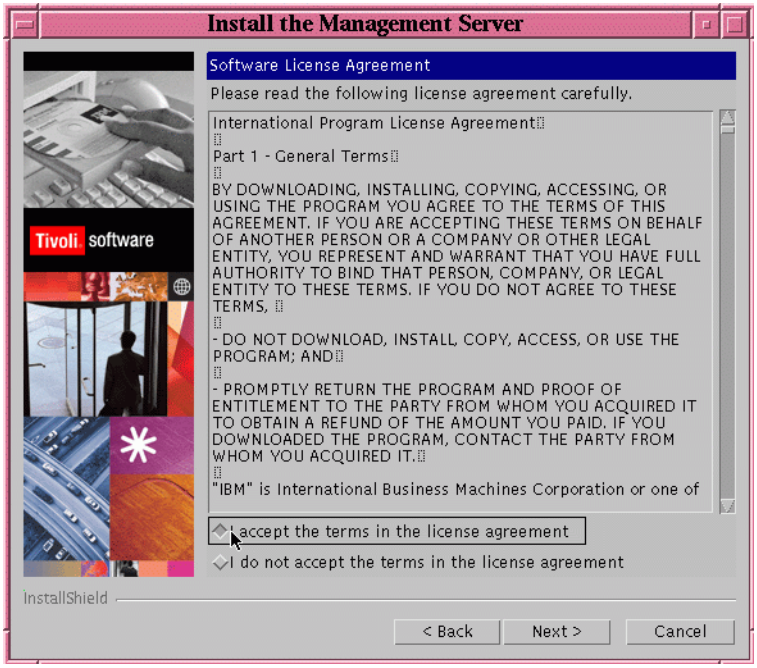

Install the Management Server
Software License Agreement
Please read the following license agreement carefully.
International Program License Agreement
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND
- PROMPTLY RETURN THE PROGRAM AND PROOF OF ENTITLEMENT TO THE PARTY FROM WHOM YOU ACQUIRED IT TO OBTAIN A REFUND OF THE AMOUNT YOU PAID. IF YOU DOWNLOADED THE PROGRAM, CONTACT THE PARTY FROM WHOM YOU ACQUIRED IT.
"IBM" is International Business Machines Corporation or one of
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 4-53 Management Server License Agreement panel

- We use the default directory to install the TMTP Management Server (Figure 4-54 on page 140).

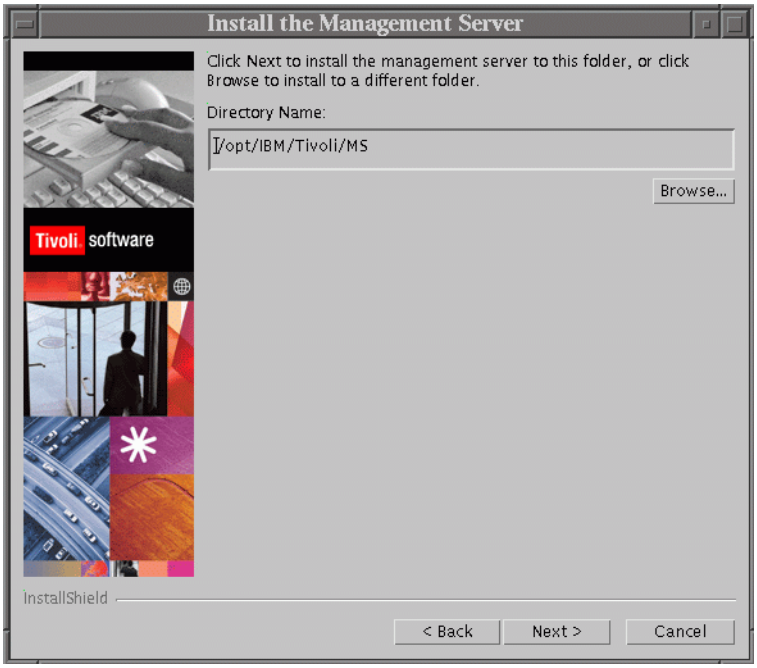


Figure 4-54 Installation location window

- Since we perform a nonsecure installation, we unchecked the **Enable SSL** option and left the port settings as the default. So the port for the non SSL agents will be 9081 and the port for the Management Server Console is set to 9082 (see Figure 4-55 on page 141).

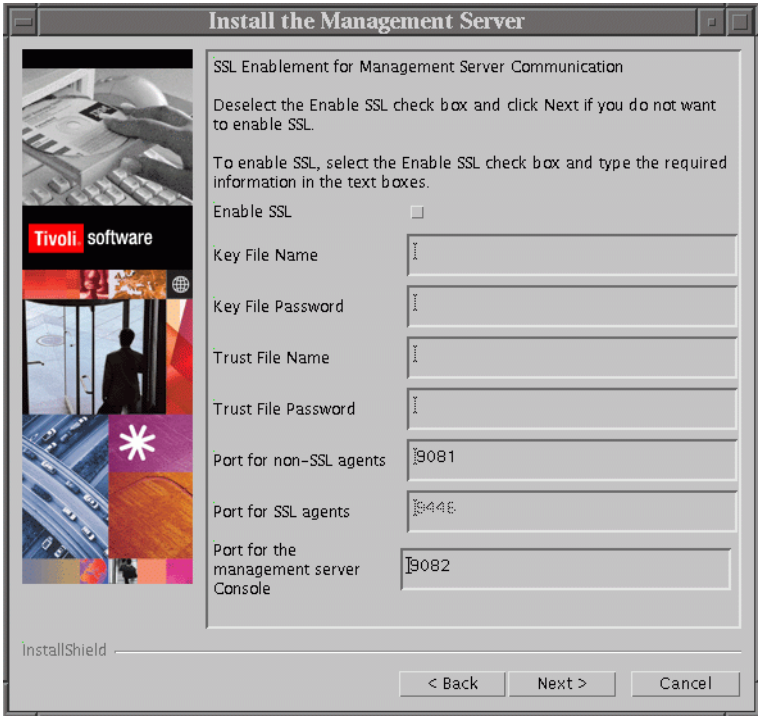

Install the Management Server
Tivoli software
SSL Enablement for Management Server Communication
Deselect the Enable SSL check box and click Next if you do not want to enable SSL.
To enable SSL, select the Enable SSL check box and type the required information in the text boxes.
Enable SSL
Key File Name
Key File Password
Trust File Name
Trust File Password
Port for non-SSL agents
9081
Port for SSL agents
9082
Port for the management server Console
9082
InstallShield
< Back
Next >
Cancel

Figure 4-55 SSL enablement window

- At the WebSphere Configuration window (Figure 4-56 on page 142), we specify the root as the user ID, which can run the WebSphere Application Server. We leave the admin console port on 9090.

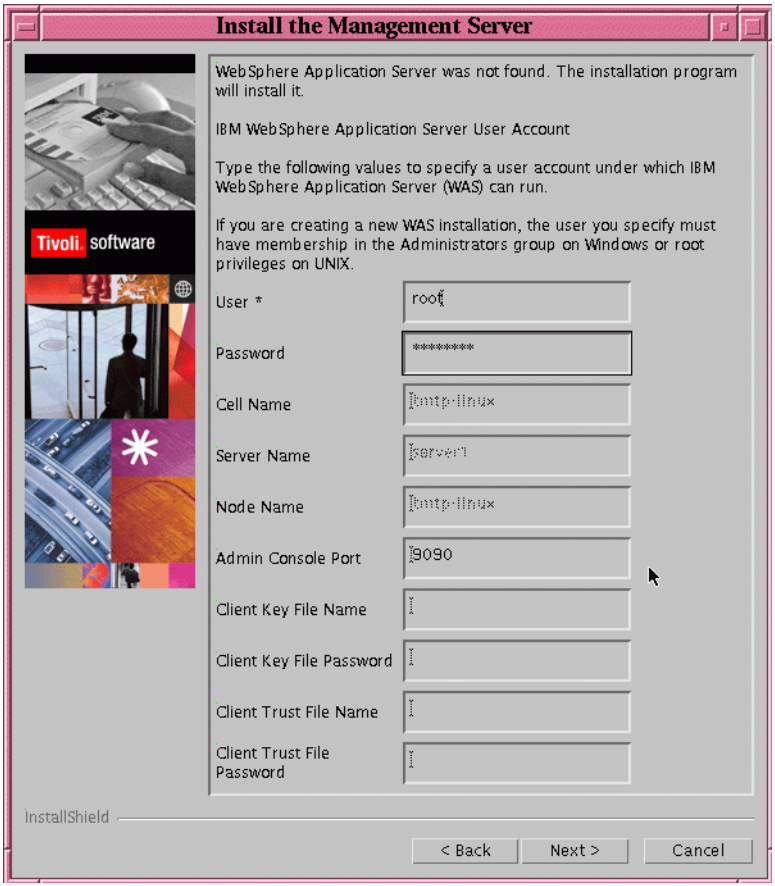

Install the Management Server
WebSphere Application Server was not found. The installation program will install it.
IBM WebSphere Application Server User Account
Type the following values to specify a user account under which IBM WebSphere Application Server (WAS) can run.
If you are creating a new WAS installation, the user you specify must have membership in the Administrators group on Windows or root privileges on UNIX.
Tivoli software
User *
root
Password
****
Cell Name
lmtp-linux
Server Name
server1
Node Name
lmtp-linux
Admin Console Port
9090
Client Key File Name
Client Key File Password
Client Trust File Name
Client Trust File Password
InstallShield
< Back
Next >
Cancel

Figure 4-56 WebSphere Configuration window

- We select the **Install DB2** option from the Database Options window (Figure 4-57 on page 143).

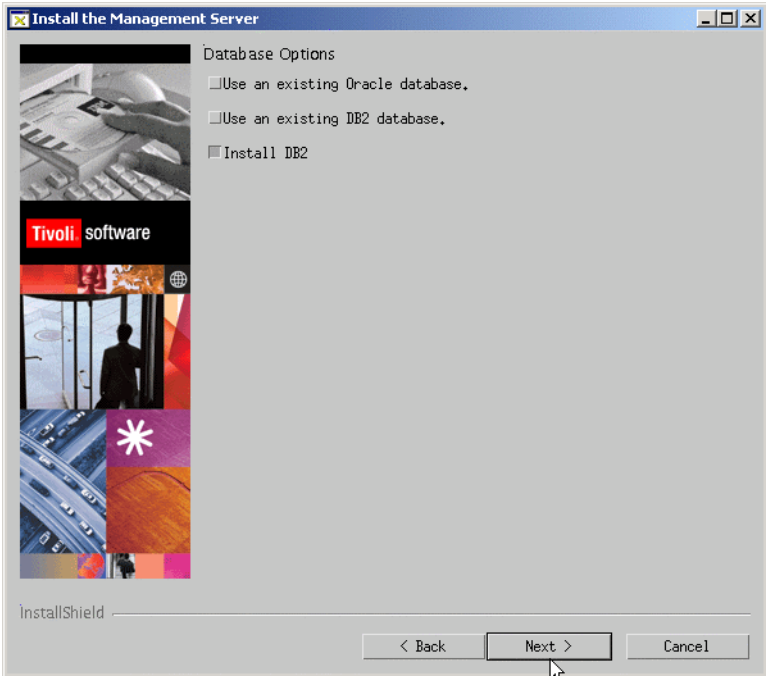


Figure 4-57 Database options window

- Figure 4-58 on page 144, we have to specify the DB2 administration account. We set this account to db2admin. We also check the **Create New User** check box so the user will be automatically created during the setup procedure.

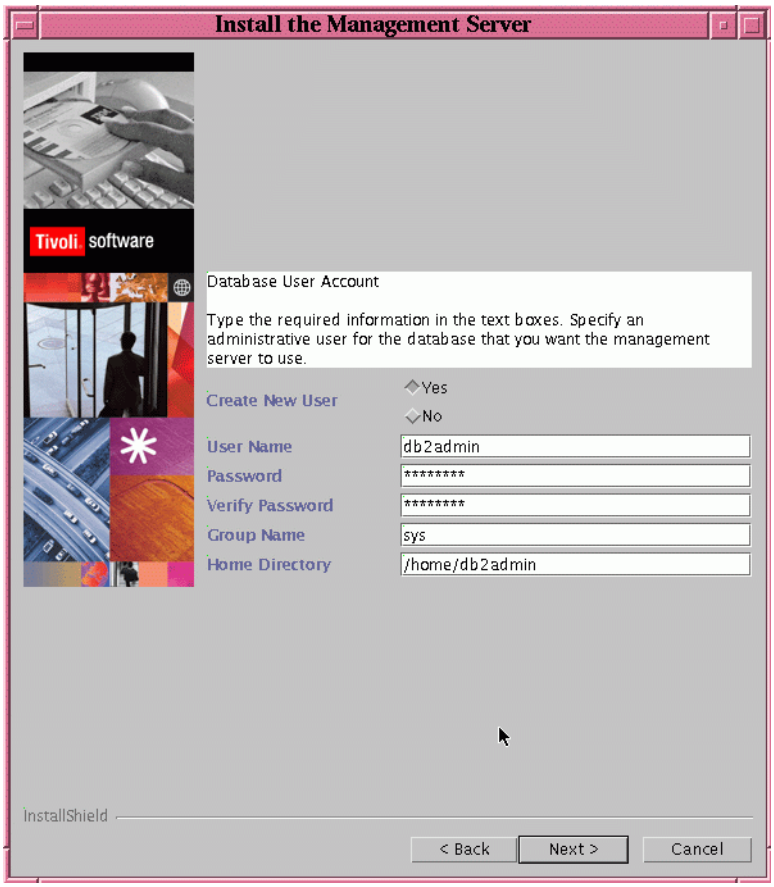


Figure 4-58 DB2 administrative user account specification

- We specify db2fenc1 as the user for the DB2 fenced operations. This is the default user (see Figure 4-59 on page 145).

Figure 4-59 User specification for fenced operations in DB2

- We specify the db2inst1 user as the DB2 instance user. The inst1 instance will hold the TMTP database (see Figure 4-60 on page 146).

Figure 4-60 User specification for the DB2 instance

- After the DB2 user is specified, the Management Server installation starts. The setup wizard copies the Management Server installation files to the specified folder, which is /opt/IBM/Tivoli/MS in this scenario (see Figure 4-61 on page 147).

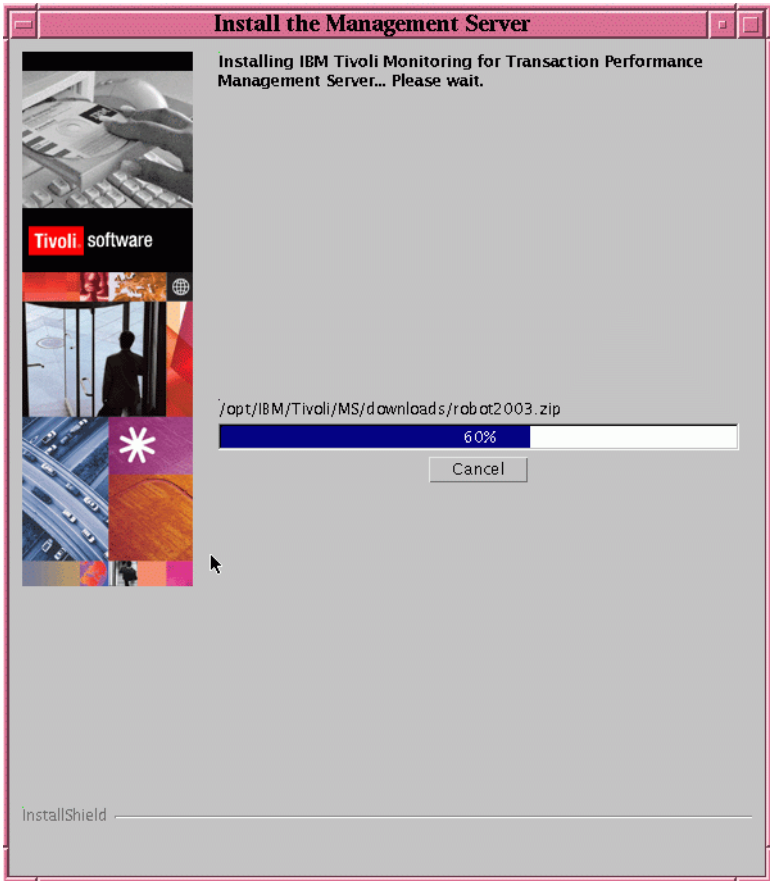


Figure 4-61 Management Server installation progress window

- Once the Management Server files are copied, the setup starts with the silent installation of the DB2 Version 8.1 server and the creation of the specified DB2 instance (see Figure 4-62 on page 148).

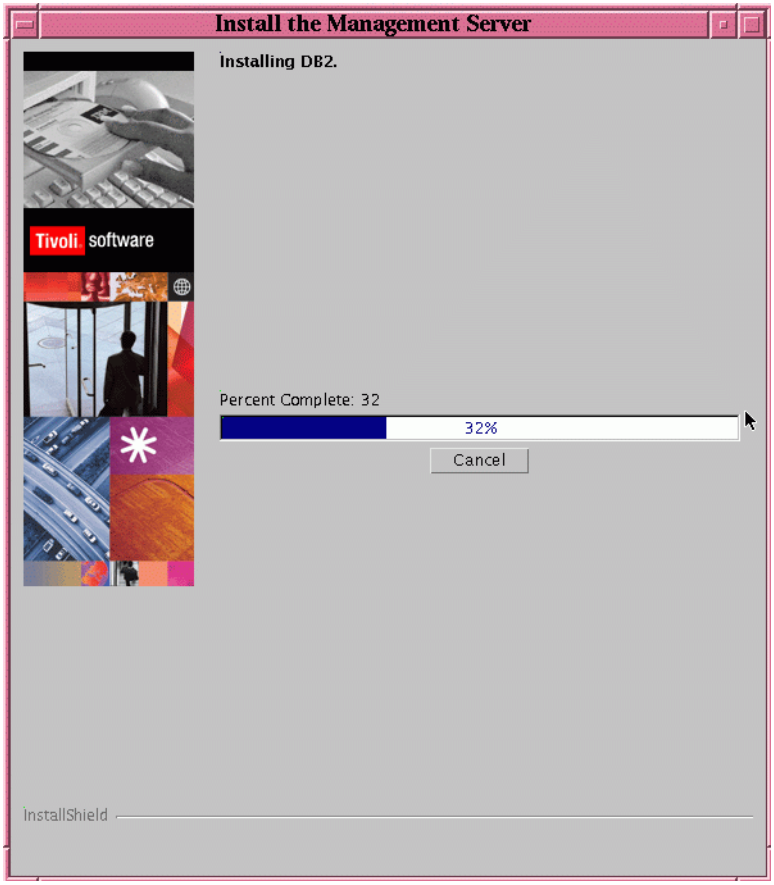


Figure 4-62 DB2 silent installation window

- When the DB2 is installed correctly, the installation wizard installs the WebSphere Application Server Version 5.0 and the WebSphere Application Server FixPack 1 (see Figure 4-63 on page 149).

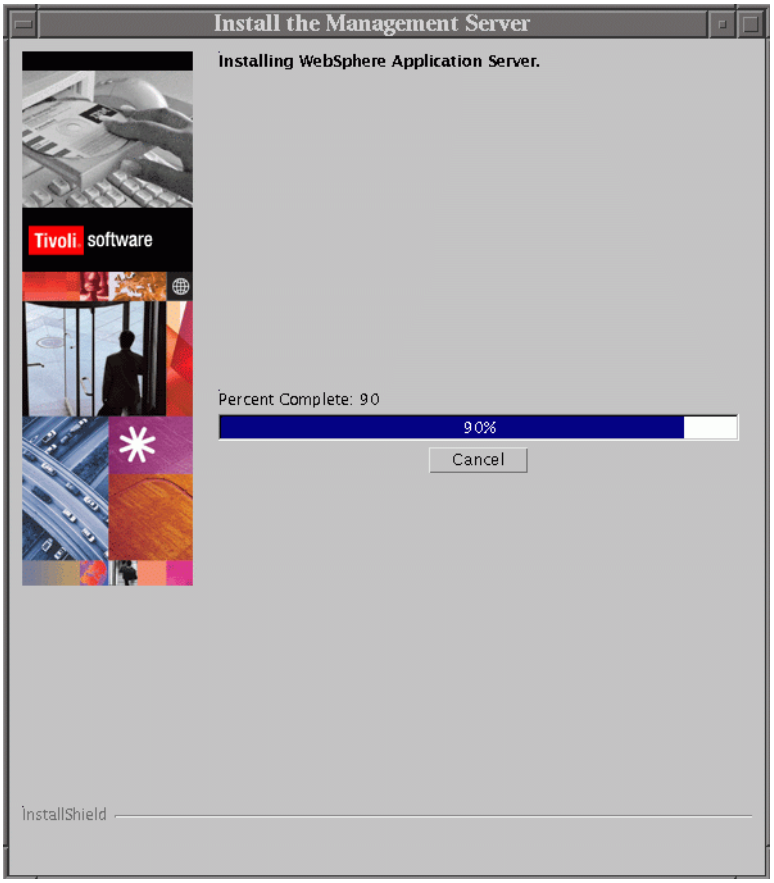


Figure 4-63 WebSphere Application Server silent installation

- If both the DB2 Version 8.1 server and the WebSphere Application Server successfully install, the setup starts creating the TMTP database and the database tables, and installs the TMTP application itself on the WebSphere Application Server (Figure 4-64 on page 150).

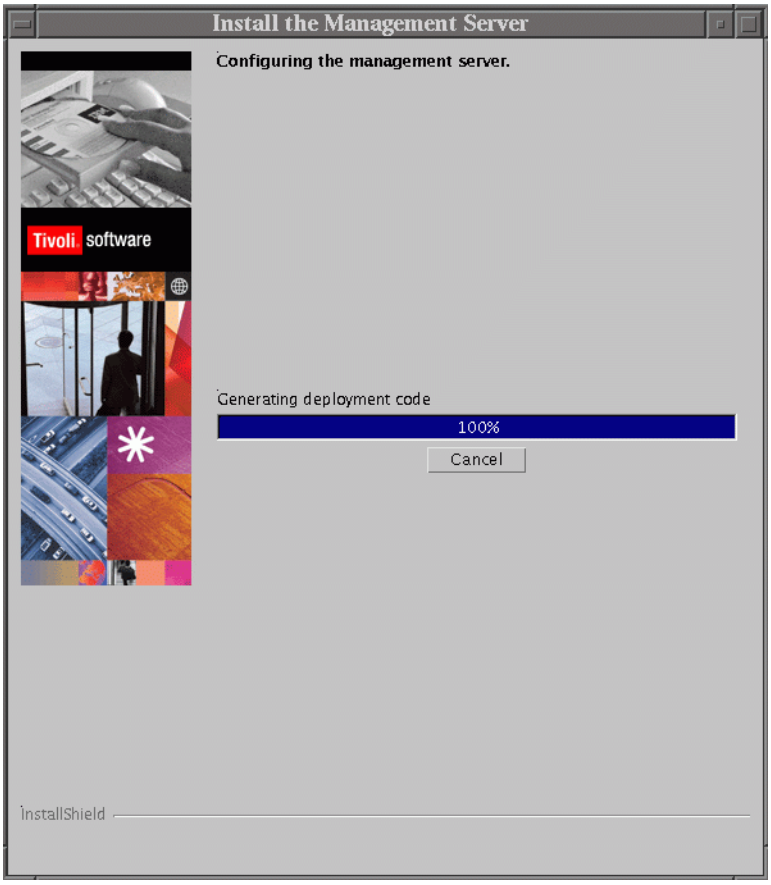


Figure 4-64 Configuration of the Management Server

- Once the installation is finished (Figure 4-65 on page 151), the WebSphere Application Server must be restarted, because the WebSphere Application Server security will now be applied. To stop and start the WebSphere Application Server, we use the following commands. These scripts are located in the $was_installation_directory/bin/. In our case, it is /opt/IBM/Tivoli/MS/WAS/bin/.

```
./stopServer.sh server1 -user root -password [password]
./startServer.sh server1 -user root -password [password]
```

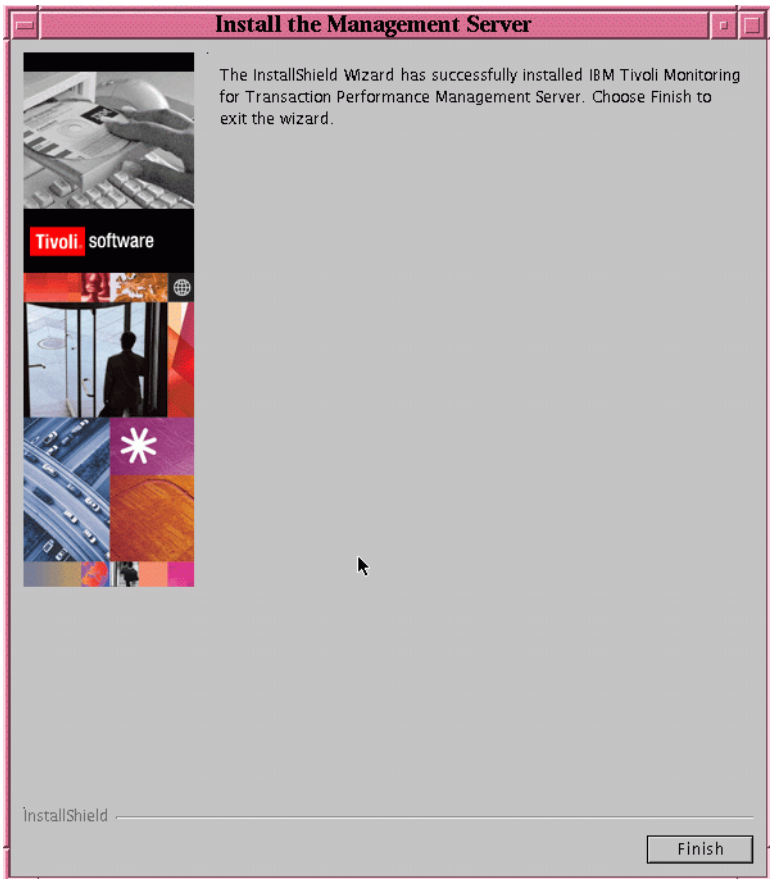

Install the Management Server
The InstallShield Wizard has successfully installed IBM Tivoli Monitoring for Transaction Performance Management Server. Choose Finish to exit the wizard.
Tivoli software
InstallShield
Finish

Figure 4-65 The finished Management Server installation

- Once the WebSphere Application Server is restarted, we log on to the TMTP server by typing the following URL into our browser:

  `http://[ipaddress]:9082/tmtpUI/`

5

Interfaces to other management tools

Every component in the e-business infrastructure is a potential show-stopper, bottleneck, or single-point-of-failure. There are a number of technologies available to allow centralized monitoring and surveillance of the e-business infrastructure components. These technologies will help manage the IT resources that are part of the e-business solution. This chapter provides a brief discussion on implementing additional Tivoli management tools that will help ensure the availability and performance of the e-business platform, as well as how to integrate TMTP with them, including integration with the following:

- Configuration of TEC to work with TMTP
- Configuration of ITM Health console to work with TMTP
- Setting SNMP
- Setting SMTP

© Copyright IBM Corp. 2003. All rights reserved.

5.1 Managing and monitoring your Web infrastructure

e-business transaction performance monitoring is important; however, it is equally important to ensure that the TMTP system itself, as well as the entire Web infrastructure, is running correctly. One of the prerequisite components for implementing TMTP is WebSphere Application Server, which in turn may rely on a prerequisite Web server, for example, IBM HTTP Server. Without these components up and running, the TMTP will not be accessible, or worse, will not work correctly. The same reason is true for the database support needed by TMTP.

The IBM Tivoli Monitoring products provide the basis for proactive monitoring, analysis, and automated problem resolution. A suite of solutions known as the "IBM Tivoli Monitoring for ..." products allow an IT department to provide management of the entire business system in a consistent way, from a central site, using an integrated set of tools.

This chapter contains multiple references to additional product documentation and other sources, such as Redbooks, which you are encouraged to refer to for further details. Please see "Related publications" on page 479 for a complete list of the referenced documents.

Note: At the time of the writing of this redbook, the publicly available version of IBM Tivoli Monitoring for Web Infrastructure does not support WebSphere Version 5.0.1. This support was being tested within IBM and was due to be released shortly after our planned publishing date.

5.1.1 Keeping Web and application servers online

The IBM Tivoli Monitoring for Web Infrastructure provides an enterprise management solution for both the Web and application server environments. The Proactive Analysis Components (PAC) that make up this product provide solutions that are integrated with other Tivoli management products. A comprehensive and fully integrated management solution can be rapidly deployed and provide a very attractive return on investment.

The IBM Tivoli Monitoring for Web Infrastructure currently focuses primarily on the performance and availability aspect of managing a Web infrastructure. The four proactive analysis components of the IBM Tivoli Monitoring for Web Infrastructure product provides similar management functions for the supported Web and application servers:

- Monitoring for IBM HTTP Server
- Monitoring for Microsoft Internet Information Server

- Monitoring for Sun iPlanet Server
- Monitoring for WebSphere Application Server

The following sections provide information on how to set up and customize IBM Tivoli Monitoring for Web Infrastructure to ensure performance and availability of the Tivoli Web Site Analyzer application.

We will focus on the monitoring for the WebSphere Application Server. For the other Web severs, refer to the redbook *Introducing IBM Tivoli Monitoring for Web Infrastructure*, SG24-6618.

5.1.2 ITM for Web Infrastructure installation

In order to install IBM Tivoli Monitoring for Web Infrastructure, you need to complete the following steps:

1. Plan your management domain.
2. Check the prerequisite software and patches.
3. Choose the installation options.
4. Verify the installation.

For all these steps, refer to the *IBM Tivoli Monitoring for Web Infrastructure Installation and Setup Guide V5.1.1,* GC23-4717 or the redbook *Introducing IBM Tivoli Monitoring for Web Infrastructure,* SG24-6618. These publications contain all the information you need to set up IBM Tivoli Monitoring for Web Infrastructure, including the prerequisites needed to install the product.

As a prerequisite to ensure the availability of TMTP, we have to ensure the availability of the WebSphere Application Server and the IBM HTTP Server.

IBM WebSphere Application Server

These are the prerequisites you need on the WebSphere Application Server system:

- IBM WebSphere Application Server Version 4.0.2 or higher.
- An operational Tivoli Endpoint.
- WebSphere Administration Server must be installed on the same system as the Tivoli endpoint.
- Java Runtime Environment Version 1.3.0 or higher.
- Monitoring at the IBM WebSphere Application Server must be enabled.

Java Runtime Environment

IBM Tivoli Monitoring for Web Infrastructure requires that the endpoints have Java Runtime Environment (JRE) Version 1.3.0 or higher installed. If a Java Runtime Environment currently is not installed on the endpoint, one can be installed from the IBM Tivoli Monitoring product CD. You can install JRE either manually or by running the **wdmdistrib -J** command, or by using the Tivoli Software Installation Service (SIS).

If you have just installed Java Runtime Environment or if you have an existing Java Runtime Environment, you need to link it to the IBM Tivoli Monitoring using the DMLinkJre task from the IBM Tivoli Monitoring Tasks TaskLibrary.

Note: For IBM WebSphere Application Server, you must use the IBM WebSphere Application Server's JRE.

Monitoring at the IBM WebSphere Application Server

The following details apply to any systems hosting IBM WebSphere Application Server that you want to manage with IBM Tivoli Monitoring for WebSphere Application Server:

- IBM Tivoli Monitoring for WebSphere Application Server supports only one installation of WebSphere Application Server on each host system.
- If security is enabled for IBM WebSphere Application Server, you should create a security properties file for the wscp client so that it can be authenticated by the server. You can copy the existing sas.client.props file in the $WAS_HOME/Properties directory ($WAS_HOME is the directory where you have installed your WebSphere Application Server) to sas.wscp.props and edit the following lines:

  ```
  com.ibm.CORBA.loginSource=properties
  com.ibm.CORBA.loginUserid=<userid>
  com.ibm.CORBA.loginPassword=<password>
  ```

  where <userid> is the IBM WebSphere Application Server user ID and <password> is the password for the user.
- If you are using a non-default port for IBM WebSphere Application Server, you need to change the configuration of the endpoint in order to communicate with the IBM WebSphere Application Server object. You can do this by changing the port setting in the sas.wscp.props file. You can create the file in the same way as mentioned above and then add the following line:

  ```
  wscp.hostPort=<port_number>
  ```

  where <port_number> is the same value specified for property com.ibm.ejs.sm.adminServer.bootstrapPort in

$WAS_HOME/bin/admin.config, where $WAS_HOME is the directory where you have installed your WebSphere Application Server.

- To monitor performance data for your IBM WebSphere administration and application servers, you must enable IBM WebSphere Application Server to collect performance data. Each performance category has an instrumentation level, which determines which counters are collected for the category. You can change the instrumentation levels using the IBM WebSphere Application Server Resource Analyzer. On the Resource Analyzer window, you need to do the following:
  - Right-click on the application server instance, for example, WebSiteAnalyzer, and choose **Properties**, click on the **Services** tab and select **Performance Monitoring Settings** from the pop-up menu to display the Performance Monitoring Settings window.
  - Select **Enable performance counter monitoring**.
  - Select a resource and choose **None, Low, Medium, High** or **Maximum** from the pop-up icon. The color associated with the chosen instrumentation level is added to the instrumentation icon and all subordinate instrumentation levels.
  - Click **OK** to apply the chosen setting or **Cancel** to undo any changes and revert to the previous setting.

  Table 5-1 lists the minimum monitoring levels for the IBM Tivoli Monitoring for Web Infrastructure WebSphere Application Server Resource Models.

Table 5-1 Minimum monitoring levels WebSphere Application Server

Resource Model	Monitoring setting	Minimum monitoring level
EJBs	Enterprise Beans	High
DB Pools	Database Connection Pools	High
HTTP Sessions	Servlet Session Manager	High
JVM Runtime	JVM Runtime	Low
Thread Pools	Thread Pools	High
Transactions	Transaction Manager	Medium
Web Applications	Web Applications	High

- You should enable the Java Virtual Machine Profile Interface (JVMPI) to improve performance analysis. The JVMPI is available on the Windows, AIX, and Solaris platforms. However, you do not need to enable JVMPI data reporting to use the Resource Models included with IBM Tivoli Monitoring for WebSphere Application Server.

IBM HTTP Server

For the prerequisites needed to monitor the IBM HTTP Server, refer to *IBM Tivoli Monitoring for Web Infrastructure Apache HTTP Server User's Guide Version 5.1*, SH19-4572.

5.1.3 Creating managed application objects

Before you start to manage Web server resources, they must first be registered in the Tivoli environment. This registration is achieved by creating specific Web Server objects in any policy region. When installing IBM Tivoli Monitoring for Web Infrastructure, a default policy region corresponding to the IBM Tivoli Monitoring for Web Infrastructure module is automatically created. For the WebSphere Application Server module, this policy region is named *Monitoring for WebSphere Application Server*.

Note: Normally, managed application objects are created in the default policy regions. If you want to create the managed application objects in a different policy region, you must first add the relevant IBM Tivoli Monitoring for Web Infrastructure managed resource to the list of resources supported by the specific policy region.

The WebSphere managed application objects are created differently from the other Web server objects. In order to manage WebSphere Application Servers, two types of WebSphere managed application objects need to be defined:

1. WebSphere Administration Server managed application object
2. WebSphere Application Server managed application object

The WebSphere Administration Server managed application object must be created before the WebSphere Application Server managed application object.

You can create the managed application object for the WebSphere Server in three different ways:

1. Using the Tivoli desktop, in which case you need to follow these two steps:
   a. Create the WebSphere Administration Server managed application object by selecting **Create** → **WSAdministrationServer** in the policy region, which will open the dialog shown in Figure 5-1 on page 159.

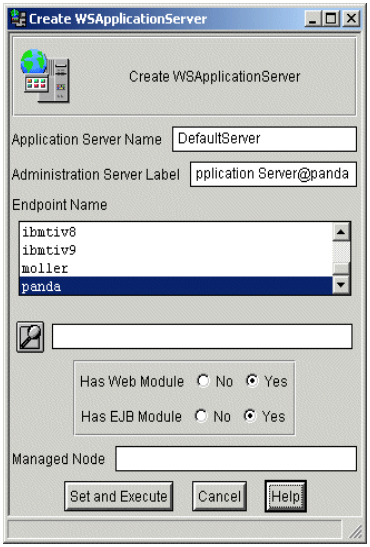


Figure 5-1 Create WSAdministrationServer

b. Create the WebSphere Application Server managed application object by selecting **Create** → **WSApplicationServer** in the policy region. The dialog in which you can specify the parameters for the managed application object is shown in Figure 5-2 on page 160.

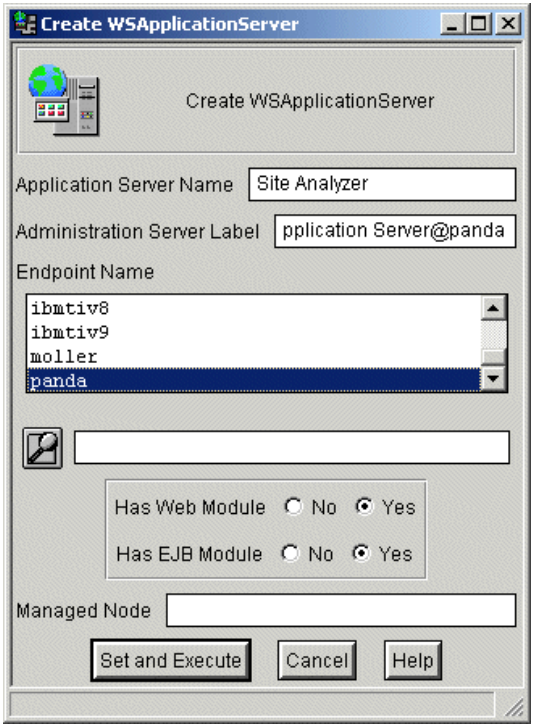


Figure 5-2 Create WSApplicationServer

2. By using the discovery task Discover_WebSphere_Resource in the TaskLibrary WebSphere Application Server Utility Tasks, both objects will be created automatically for you. When starting the task, supply the parameters for discovery in the dialog, as shown in Figure 5-3 on page 161.

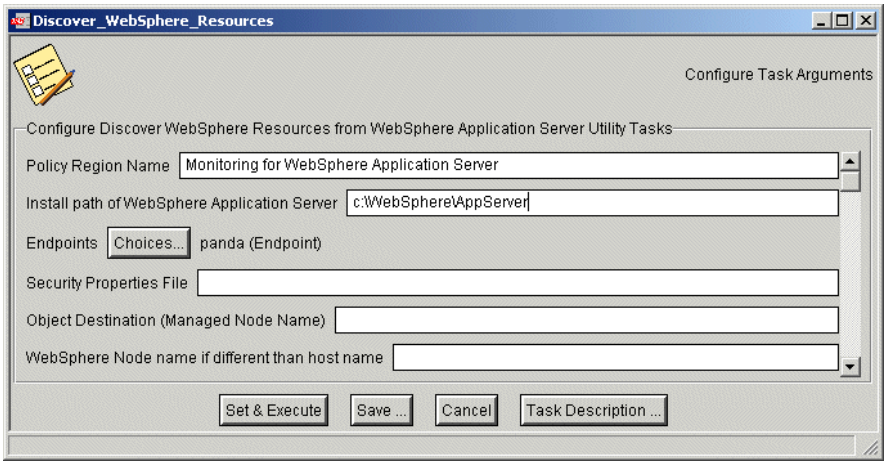


Figure 5-3 Discover WebSphere Resources

3. Run the appropriate command from the command line:

   ```
   wWebshpere -c
   ```

   Note: This method can only be used to create the WebSphere Application Server managed application object.

   For all the specified parameters, commands, and the appropriate descriptions, refer to the *IBM Tivoli Monitoring for Web Infrastructure Reference Guide Version 5.1.1,* GC23-4720 and the *IBM Tivoli Monitoring for Web Infrastructure: WebSphere Application Server User's Guide Version 5.1.1,* SC23-4705.

If all the parameters supplied to the Tivoli Desktop, the command line, or the task are correct, the managed server objects icons shown in Figure 5-4 on page 162 are added to the policy region.

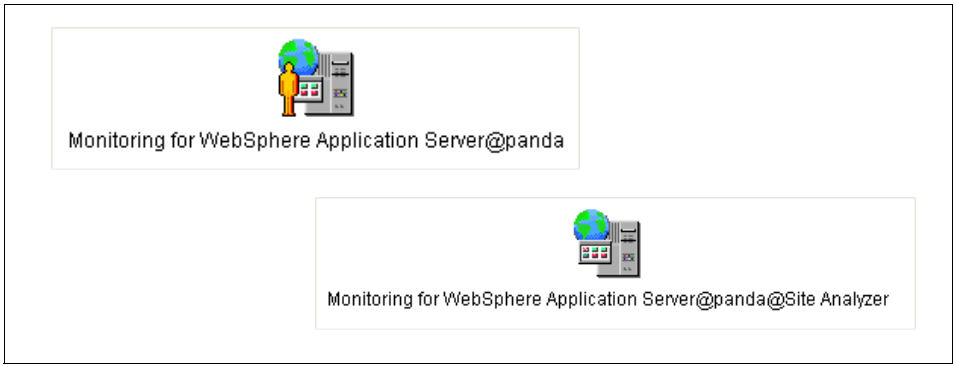

Monitoring for WebSphere Application Server@panda
Monitoring for WebSphere Application Server@panda@Site Analyzer

Figure 5-4 WebSphere managed application object icons

5.1.4 WebSphere monitoring

The following section will outline tasks needed to activate monitoring of the availability and performance of the Tivoli Web Site Analyzer application's operational environment with IBM Tivoli Monitoring for Web Infrastructure.

Resource Models

A Resource Model is used to monitor, capture, and return information about multiple resources and applications. When adding Resource Models to a profile, these are chosen based on the type of resources that are being monitored.

WebSphereAS is the abbreviated name of the IBM Tivoli Monitoring category of the IBM WebSphere Application Server Resource Models. It is used as an identifying prefix.

Planning

The following list gives the indicators available in the Resource Models provided with the Tivoli PAC for WebSphere Application Server:

- WebSphereAS Administration Server Status: Administration server is down, occurs when the status of the WebSphere Application Server administration server is down.
- WebSphereAS Application Server Status: Application server is down, occurs when the status of the WebSphere Application Server application server is down.

- WebSphereAS DB Pools:
  - Connection pool timeouts are too high, which occur when the database connection timeout exceeds a predefined threshold.
  - DB Pools avgWaitTime is too high, which occurs when the average time required to obtain a connection in the database connection pool exceeds the predefined threshold.
  - Percent connection pool used is too high, which occurs when the percentage of database connection in use is higher than a predefined threshold (assuming you have sufficient network capacity and database availability, you might need to increase the size of the database connection pool).
- WebSphereAS JEJB:
  - Enhanced Java Bean (EJB) performance, either gathered at the EJB or application server (EJB container) level, which occurs when the average method response time (ms) exceeds the response time threshold. The load is also reported by concurrent active EJB requests, and throughput is measured by the EJB request rate per minutes.
  - EJB exceptions, either gathered at the EJB or application server (EJB container) level, which occur when a specified percentage of EJBs are being discarded instead of returned to the pool. The returns discarded (as a percentage of those returned to the pool) exceeded the defined threshold. If you receive this indication, you may need to increase the size of your EJB pool.
- WebSphereAS HTTP Sessions: LiveSessions is too high, which occurs when the number of live sessions exceeds the predefined “normal” amount for an application.
- WebSphereAS JVM Runtime: Used JVM memory is too high, which occurs when the percentage of used JVM memory exceeds a defined percentage of the total available memory.
- WebSphereAS Thread Pools: Thread pool load, which occurs when the ratio of active threads to the size of the thread pool exceeds the predefined threshold.
- WebSphereAS Transaction:
  - The recent transaction response time is too high, which occurs when the average transaction response time exceeds a predefined threshold.
  - The timed-out transactions are too high, which occur when transactions exceed the time-out limit and are being terminated (a maximum ratio for timed-out transactions to total transactions).

- WebSphereAS Web Applications:
  - Servlet/JSP errors, either at the application server or Web application or servlet level, which occurs when the number of servlet error passes a predefined *normal* amount of errors for the application.
  - Servlet/JSP performance, either at the application server or Web application or servlet level, which occurs when the servlet response time exceeds the predefined monitoring threshold.

During the initial deployment on any Resource Model of IBM Tivoli Monitoring for Web Infrastructure, we recommend using the default values shown in Table 5-2. The following definitions will help you understand the table.

Number of Occurrences
Specifies the number of consecutive times the problem occurs before the software generates an indication.

Number of Holes Determines how many cycles that do not produce an indication can occur between cycles that do produce an indication.

Table 5-2 Resource Model indicator defaults

	Indication	Cycle time	Threshold	Occurrences /Holes
WebSphereAS Administration Server Status				
	Administration Server is down.	60s	down	1/0
WebSphereAS Application Server Status				
	Application Server is down.	60s	down	1/0
WebSphereAS DB Pools				
	Connection pool timeouts are too high.	90s	0	9/1
	DB Pool avgWaitTime is too high.	90s	250ms	9/1
	Percent connection Pool used is too high.	90s	90	9/1
WebSphereAS EJB				
	EJB performance (data gathered at EJB level).	90	0	9/1
	EJB performance (data gathered at application server, EJB container, and level).	90	0	9/1

	Indication	Cycle time	Threshold	Occurrences /Holes
	EJB exceptions (data gathered at EJB level).	90s	50%	9/1
	EJB exceptions (data gathered at application server, EJB container, and level).	90s	50%	9/1
WebSphereAS HTTP Sessions				
	LiveSessions is too high.	180s	1000	9/1
WebSphereAS JVM Runtime				
	Used JVM memory is too high.	60s	95%	1/0
WebSphereAS Thread Pools				
	Thread Pool load.	180s	95%	9/1
WebSphereAS Transactions				
	Recent transaction response time is too high.	180s	1000ms	9/1
	Timed-out transactions are too high.	180s	2%	9/1
WebSphereAS Web Applications				
	Servlet/JSP errors (at application server level).	90s	0	9/1
	Servlet/JSP errors (at Web application level.	90s	0	9/1
	Servlet/JSP errors (at servlet level).	90s	0	9/1
	Servlet/JSP performance (at application server level).	90s	750ms	9/1
	Servlet/JSP performance (at Web application level.	90s	750ms	9/1
	Servlet/JSP performance (at servlet level).	90s	750ms	9/1

Deployment

After deciding which Resource Models and indications you need, you have to deploy the monitors. This means you have to:

1. Create profile managers and profiles. This will help organize and distribute the Resource Models.

   A monitoring profile may be regarded as a group of customized Resource Models that can be distributed to a managed resource in a profile manager. The profile manager has to be created first with the **wcrtprfmgr** command or from the Tivoli desktop. After this, you can create the profile, which should be a Tmw2kProfile (must be included in the managed resources of the policy region), with the **wcrtprf** command or from the Tivoli desktop.

2. Add subscribers to the profile managers.

   The subscribers of a profile manager determine which systems will be monitored when the profile is distributed. You can do this with either the **wsub** command or from the Tivoli desktop. The subscribers for IBM Tivoli Monitoring for Web Infrastructure would be the managed application objects that were created in 5.1.3, “Creating managed application objects” on page 158.

3. Add Resource Models.

   We recommend that you group all of the Resource Models to be distributed to the same endpoint or managed application object in a single profile. You can now add the Resource Models with the parameters you have chosen to the profiles. You can do this by using either the **wdmeditprf** command or the Tivoli desktop, as shown in Figure 5-5 on page 167.

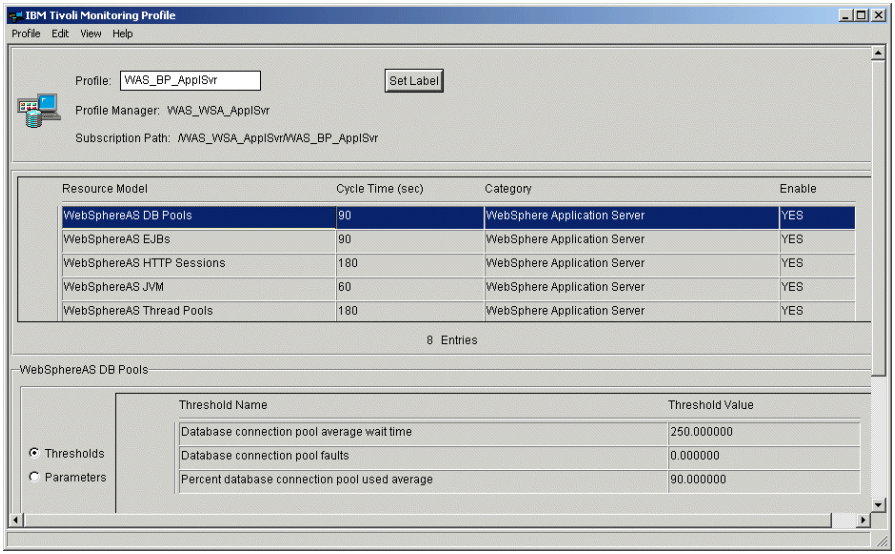


Figure 5-5 Example for an IBM Tivoli Monitoring Profile

4. Distribute the profiles.

   You can do this by either using the **wdmdistrib** command or the Tivoli desktop.

Tivoli Enterprise Console adapter

By default, all the Resource Models will send an event to the Tivoli Event Console event management environment whenever a threshold is violated. These events may be used to trigger actions based on rules stored in the TEC Server.

Another possible way to send events to the TEC environment is directly from the WebSphere Application Server using the IBM WebSphere Application Server Tivoli Enterprise Console adapter. This adapter is used to forward native WebSphere Application Server messages (SeriousEvents) to the Tivoli Enterprise Console. These messages may have the following severity codes:

- FATAL
- ERROR
- AUDIT
- WARNING
- TERMINATE

The Tivoli Enterprise Console adapter is also self-reporting; you can see adapter status events in the WebSphere Application Server console.

A task is created during the installation of the product in the WebSphere Event Tasks TaskLibrary. This task, Configure_WebSphere_TEC_Adapter, is used to configure the adapter. Before executing this task, make sure that the IBM WebSphere Administration Server is running. Then you have to configure which messages you want to be forwarded to the Tivoli Enterprise Console.

The WebSphere Event Tasks TaskLibrary also includes two tasks with which you can start and stop the Tivoli Enterprise Console adapter. The task names are:

- Start_WebSphere_TEC_Adapter
- Stop_WebSphere_TEC_Adapter

5.1.5 Event handling

Tivoli Enterprise Console (TEC) has been designed to receive events from multiple sources and process them in order to correlate and aggregate them, and issue predefined (corrective) actions based on the processing. TEC works on the basis of events and rules.

TEC events are defined in object-oriented definition files called BAROC files. These events are defined hierarchically according to their type. Each event type is called an event class. When TEC receives an event, it parses the event to determine the event class and then apply the class definition to parse the rest of the event; when the parsing is successful, the event is stored in the TEC database.

When a new event is stored, a timer expires, or a field (known in TEC terminology as a slot) has changed, TEC evaluates a set of rules to be applied to the event. These rules are stored in ruleset files, which are written in the Prolog language. When a matching rule is found, the action part of the rule is executed. These rules enable events to be correlated and aggregated. Rules also enable automatic responses to certain conditions; usually, these are corrective actions.

In the IBM Tivoli Monitoring for Web Infrastructure perspective, Web- and application server specific events are generated by the Resource Models provided by each of the IBM Tivoli Monitoring for Web Infrastructure modules. These events are defined in TEC and a set of predefined rules exists to correlate and process the events.

To set up a TEC environment capable of receiving Web and application server related events from IBM Tivoli Monitoring for Web Infrastructure environment, at least the following components have to be installed:

- Tivoli Enterprise Console Server Version 3.7.1
- Tivoli Enterprise Console Version 3.7.1
- Tivoli Enterprise Console User Interface Server Version 3.7.1

- Tivoli Enterprise Console Adapter Configuration Facility Version 3.7.1

TEC also uses a RDBMS system in which events are stored. Please refer to the *IBM Tivoli Enterprise Console User's Guide Version 3.8,* GC32-0667 for further details on TEC installation and use.

IBM Tivoli Monitoring for Web Infrastructure events and rules

In order to define the IBM Tivoli Monitoring for Web Infrastructure related events and rules to the TEC, the proper definition files have to be imported into the TEC environment. The IBM Tivoli Monitoring for Web Infrastructure events and rules are described in files that have .baroc and .rls file extensions. All the files can be found in the directory in which the Tivoli Enterprise Console server code is installed (in the subdirectory bin/generic_unix/TME®).

The definition files for the IBM Tivoli Monitoring for WebSphere Application Server events are documented in the subdirectory WSAPPSVR in the following BAROC files:

itmwas_dm_events.baroc
: Definitions for the events originated from all the Resource Models

itmwas_events.baroc
: Definitions of events forwarded to the TEC directly from the WebSphere Application Server and the Tivoli Enterprise Console adapter

For the IBM Tivoli Monitoring for WebSphere Application Server events, three different rulesets are supplied in the subdirectory WSAPPSVR:

itmwas_events.rls
: Handles events that originate directly from the WebSphere Application Server Tivoli Enterprise Console adapter

itmwas_monitors.rls
: Handles events that originate from Resource Models

itmwas_forward_tbsm.rls
: Handles events that are forwarded to Tivoli Business System Manager

Tivoli provides for all the IBM Tivoli Monitoring for Web Infrastructure solutions definition files and ruleset files. They are located in the appropriate subdirectories. For documentation regarding these files, please refer to the appropriate User's Guides for the IBM Tivoli Monitoring for Web Infrastructure modules.

For further information on how to implement the classes and rule files, refer to the *IBM Tivoli Enterprise Console Rule Builder's Guide Version 3.8,* GC32-0669.

5.1.6 Surveillance: Web Health Console

You can use the IBM Tivoli Monitoring Web Health Console to display, check, and analyze the status and health of any endpoint, where monitoring has been activated by distributing profiles with Resource Models. The endpoint status reflects the state of the endpoint displayed on the Web Health Console, such as running or stopped. Health is a numeric value determined by Resource Model settings. The typical settings include required occurrences, cycle times, thresholds, and parameters for indications. These are defined when the resource model is created. You can also use the Web Health Console to work with real-time or historical data from an endpoint that is logged to the IBM Tivoli Monitoring database.

You can connect the Web Health Console to any Tivoli management region server or managed node and configure it to monitor any or all of the endpoints that are found in that region. The Web Health Console does not have to be within the region itself, although it may.

To connect to the Web Health Console, you need access to the server on which the Web Health Console server is installed and the Tivoli Management Region on which you want to monitor the Health Console. All user management and security is handled through the Tivoli management environment. This includes creating users and password as well as assigning authority.

To activate the online monitoring of the health of a resource, you have to log in to the Web Health Console. This may be achieved by performing the following steps:

1. Open your browser and type the following text in the address field:

   `http://<server_name>/dmwhc`

   where <server_name> is the fully qualified host name or IP address of the server hosting the Web Health Console.

2. Supply the following information:

   User Tivoli user ID

   Password Password associated with the Tivoli user ID

   Host name The managed node to which you want to connect

3. The first time you log in to the Web Health Console, the Preferences view is displayed. You must populate the Selected Endpoint list before you can access any other Web Health Console views. When you log in subsequently, the endpoint list is loaded automatically.

4. Select the endpoints that you want to monitor and choose the Endpoint Health view. This is the most detailed view of the health of an endpoint. In this view, the following information is displayed:
   a. The health and status of all Resource Models installed on the endpoint.
   b. The health of the indications that make up the Resource Model and historical data.

After setting up the Web Health Console, you are able to display the health of a specific endpoint; to view the data, use the theoretical view option. Figure 5-6 shows an example of real-time monitoring of an WebSphere Application Server.

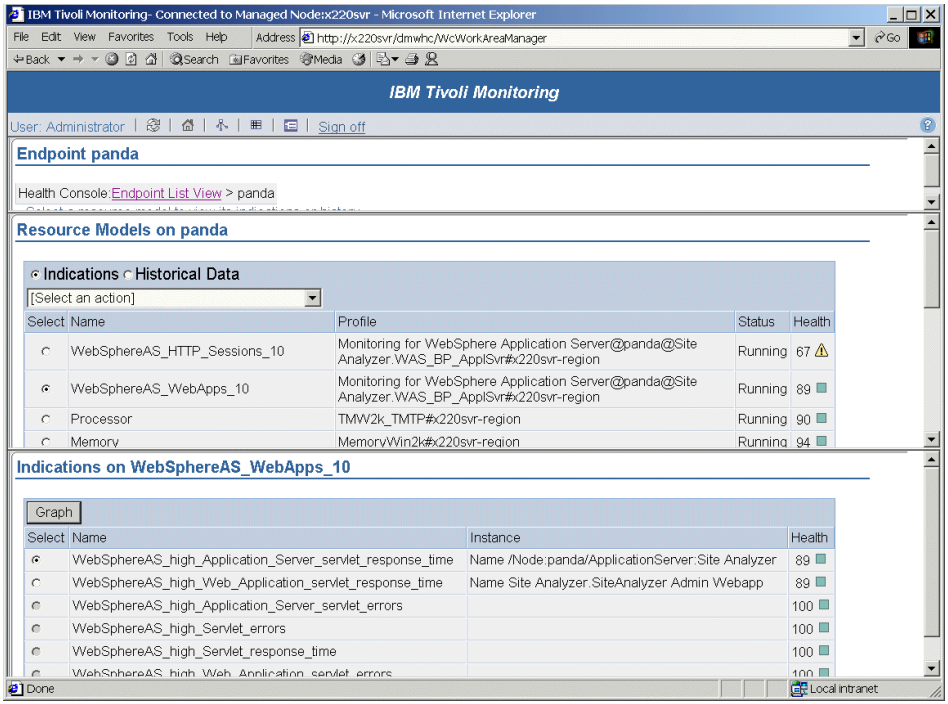

IBM Tivoli Monitoring- Connected to Managed Node:x220svr - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://x220svr/dmwhc/WcWorkAreaManager
IBM Tivoli Monitoring
User: Administrator
Sign off
Endpoint panda
Health Console:Endpoint List View > panda
Resource Models on panda
Indications Historical Data
[Select an action]
Select Name Profile Status Health
WebSphereAS_HTTP_Sessions_10 Monitoring for WebSphere Application Server@panda@Site Analyzer:WAS_BP_ApplSvr#x220svr-region Running 67
WebSphereAS_WebApps_10 Monitoring for WebSphere Application Server@panda@Site Analyzer:WAS_BP_ApplSvr#x220svr-region Running 89
Processor TMW2k_TMTP#x220svr-region Running 90
Memory MemoryWin2k#x220svr-region Running 94
Indications on WebSphereAS_WebApps_10
Graph
Select Name Instance Health
WebSphereAS_high_Application_Server_servlet_response_time Name /Node:panda/ApplicationServer:Site Analyzer 89
WebSphereAS_high_Web_Application_servlet_response_time Name Site Analyzer:SiteAnalyzer Admin Webapp 89
WebSphereAS_high_Application_Server_servlet_errors 100
WebSphereAS_high_Servlet_errors 100
WebSphereAS_high_Servlet_response_time 100
Done
Local intranet

Figure 5-6 Web Health Console using WebSphere Application Server

For detailed information on setting up and working with the Web Health Console, refer to the *IBM Tivoli Monitoring User's Guide V5.1.1*, SH19-4569.

5.2 Configuration of TEC to work with TMTP

Follow these steps to configure TMTP to forward events to TEC:

1. Navigate to the MS/config/ directory.

2. Locate the eif.conf file. In the eif.conf file, define the TEC server by setting the ServerLocation property to the name of the Management Server (see Example 5-1).

Example 5-1 Configure TEC

```
#The ServerLocation keyword is optional and not used when the TransportList
keyword
#is specified.
#
#Note:
#    The ServerLocation keyword defines the path and name of the file for
logging
#events, instead of the event server, when used with the TestMode keyword.
##############################################################################
#
# NOTE: SET THE VALUE BELOW AS SHOWN IN THIS EXAMPLE TO CONFIGURE TEC EVENTS
#
# Example: ServerLocation=marx.tivlab.austin.ibm.com
#
ServerLocation=<your_fully_qualified_host_name_goes_here>

##############################################################################
#ServerPort=number
#
#Specifies the port number on a non-TME adapter only on which the event server
#listens for events. Set this keyword value to zero (0), the default value,
#unless the portmapper is not available on the event server, which is the case
#if the event server is running on Microsoft Windows or the event server is a
#Tivoli Availability Intermediate Manager (see the following note). If the port
#number is specified as zero (0) or it is not specified, the port number is
#retrieved using the portmapper.
#
#The ServerPort keyword is optional and not used when the TransportList keyword
#is specified.
##############################################################################
ServerPort=5529
```

3. Set the port number for the Management Server.
4. Shut down and restart WebSphere Application Server on the management server system. To shut down and restart WebSphere Application Server, use the **stopserver <servername>** command located in the WebSphere/AppServer/bin directory.

5.2.1 Configuration of ITM Health Console to work with TMTP

Use the User Settings window shown in Figure 5-7 on page 174to change any of the following optional settings:

- Time zone shown for time stamps in the user interface.
- Web Health Console usernames, passwords and server.

  This information enables IBM Tivoli Monitoring for Transaction Performance to connect to the Web Health Console. The Tivoli Web Health Console presents monitoring data for those IBM Tivoli Monitoring products that are based on resource models. For example, the Web Health Console displays data captured by products such as IBM Tivoli Monitoring for Databases and IBM Tivoli Monitoring for Business Integration.
- Refresh rate for the Web Health Console display.

  Keep the default refresh rate of five minutes or change it according to your needs.
- Configure the Time Zone performing the following steps:

  a. Select a time zone from the Time Zone drop-down list.

  b. Place a check mark in the box to enable automatic adjustment for Daylight Savings Time.

  c. Provide the following information regarding the environment of the Web Health Console:

  - Type the following information about the Tivoli managed node (also referred to as the TME) that is monitoring server endpoints:

    TME Host name: The fully qualified host name or the IP address of the Tivoli managed node.

    Additional Information: The host that you specify for the Tivoli managed node might be the same computer that hosts the Tivoli management region server. This sharing of the host computer might exist in smaller Tivoli environments, for example, when Tivoli is monitoring fewer than 10 endpoints. When the Tivoli environment monitors hundreds of endpoints, the host for the Tivoli managed node is likely to be different from the host for the Tivoli management region server.

    Note: Do not include the protocol in the host name. For example, type myserver.ibm.tivoli.com, not http://myserver.ibm.tivoli.com.

    TME Username: Name of a valid user account on the host computer.

    TME Password: Password of the user account on the host computer.

- Type the following information about the Integrated Solutions Console (also referred to as the ISC):

  Additional Information: The Integrated Solutions Console is the portal for the Web Health Console. These consoles run on an installation of the WebSphere Application Server.

  ISC Username: Name of a valid user account on the computer for the Integrated Solutions Console.

  ISC Password: Password of the user account.

- Type the Internet address of the Web Health Console server in the WHC Server text box in the following format:

  `http://host_computer_name/LaunchITM/WHC`

  where *host_computer_name* is the fully qualified host name for the computer that hosts the Web Health Console.

Note: The Web Health Console is a component that runs on an installation of WebSphere Application Server.

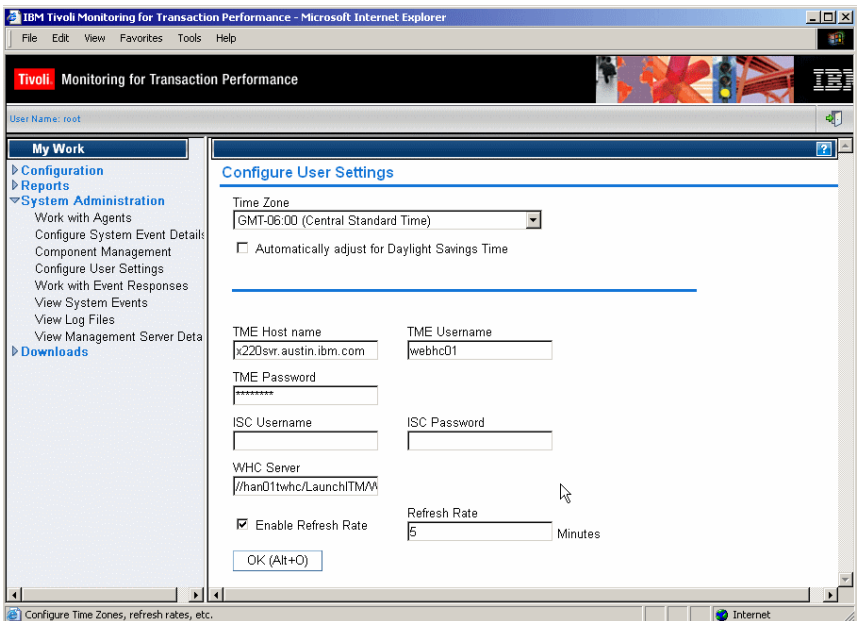


Figure 5-7 Configure User Setting for ITM Web Health Console

Configure the refresh rate for the Web Health Console as follows:

1. Select the **Enable Refresh Rate option** to override the default refresh rates for the Web Health Console display.
2. Type an integer in the **Refresh Rate field** to specify the number of minutes that pass between each refresh.
3. Click **OK** to save the user settings and enable connection to the Web Health Console.

5.2.2 Setting SNMP

Set SNMP by following these steps:

1. Open the <MS_Install_Dir >/config directory, where <MS_Install_Dir> is the directory containing the Management Server installation files.
2. Open the tmtp.properties property file.
3. Modify the EventService.SNMPServerLocation key with the fully-qualified server name, such as EventService.SNMPServerLocation=bjones.austin.ibm.com.
4. (Optional) Modify the EventService.SNMPPort key to specify a different port number than the default value of 162.
5. (Optional) Modify the SMTPProxyPort key to specify a fully-qualified proxy server host name.
6. (Optional) Modify the EventService.SNMPV1ApiLogEnabled key to enable debug tracing in the classes found in the snmp.jar file.

   Additional Information: The output produced by this tracing writes to the WebSphere log files found in <WebSphere_Install_Dir>/WebSphere/AppServer/logs/<server_name>, where <WebSphere_Install_Dir> is the name of the WebSphere Installation Directory and <server_name> is the name of the server.

7. Perform one of the following actions to complete the procedure:
   - Restart WebSphere Application Services.
   - Restart the IBM Tivoli Monitoring for Transaction Performance from the WebSphere administration console.

5.2.3 Setting SMTP

Set SMTP by following these steps:

1. Open the <MS_Install_Dir >/config directory, where <MS_Install_Dir> is the name of the Management Server directory.
2. Open the tmtp.properties property file.
3. Modify the SMTPServerLocation key with the fully-qualified SMTP server host name.

   Additional Information: The host name is combined with the domain name, for example, my_hostname.austin.ibm.com.

4. (Optional) Modify the SMTPProxyHost key to specify a fully-qualified proxy server host name.
5. (Optional) Modify the SMTPProxyPort key to specify a port number other than the default value.
6. (Optional) Modify the SMTPDebugMode key to enable debug tracing in the classes found in the mail.jar file when the value is set to true.

   Additional Information: Trace information can help resolve problems with e-mail.

7. Perform one of the following actions to complete the procedure:
   - Restart WebSphere Application Services.
   - Restart the IBM Tivoli Monitoring for Transaction Performance from the WebSphere administration console.

6

Keeping the transaction monitoring environment fit

This chapter describes some general maintenance procedures for TMTP Version 5.2 including:

- How to start and stop various components.
- How to uninstall the Management Server cleanly from a UNIX® platform.

We also describe some of the configuration options and provide the reader with some general troubleshooting procedures.

Lastly, we discuss using various other IBM Tivoli products to manage the availability of the TMTP application.

The TMTP product includes a comprehensive manual for troubleshooting; this chapter does not attempt to reproduce that information.

© Copyright IBM Corp. 2003. All rights reserved.

6.1 Basic maintenance for the TMTP WTP environment

- The TMTP WTP environment is based on the DB2 database sever and the WebSphere 5.0 Application Server, so it is important to understand some basic maintenance tasks related to these two products.
- To stop and start the DB2 Database Server open a DB2 command line processor window and type the following commands:

  ```
  db2stop
  db2start
  ```

  The database log file can be found at /instance_home/sqllib/db2dump/db2diag.log.

  Tip: Our recommendation is to use a tool, such as IBM Tivoli Monitoring for Databases, to monitor the following TMTP DB2 parameters**:**

  - DB2 Instance Status
  - DB2 Locks and Deadlocks
  - DB2 Disk space usage

- To stop and start the WebSphere Application Server, type the following commands:

  ```
  ./stopServer.sh server1 -user root -password [password]
  ./startServer.sh server1 -user root -password [password]
  ```

  The WebSphere application server logs can be found under the following directories:

  – [WebSphere_installation_folder]/logs/

  – [WebSphere_installation_folder]/logs/[servername]/

  Important: Prior to starting WebSphere on a UNIX platform, you will need to source the DB2 environment. This can be done by sourcing the db2profile script from the home directory of the relevant instance user id. For us, the command for this was **`. /home/db2inst1/sqllib/db2profile`**. If this is not done, you will receive JDBC errors when trying to access the TMTP User Interface via a Web Browser (see Figure 6-1 on page 179).

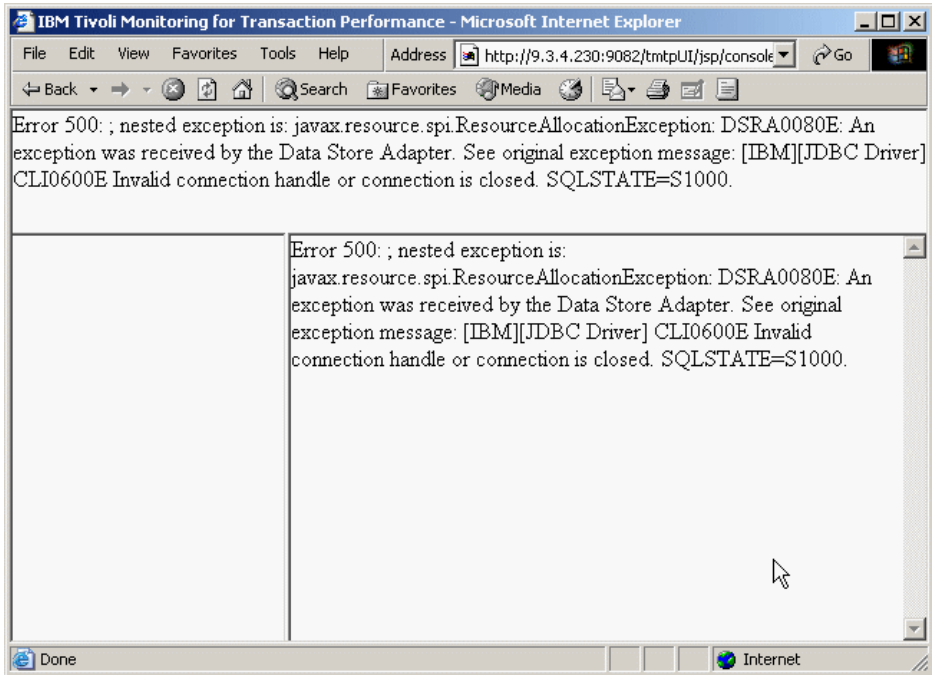

IBM Tivoli Monitoring for Transaction Performance - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://9.3.4.230:9082/tmtpUI/jsp/console
Go
Back Search Favorites Media
Error 500: ; nested exception is: javax.resource.spi.ResourceAllocationException: DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: [IBM][JDBC Driver] CLI0600E Invalid connection handle or connection is closed. SQLSTATE=S1000.
Error 500: ; nested exception is: javax.resource.spi.ResourceAllocationException: DSRA0080E: An exception was received by the Data Store Adapter. See original exception message: [IBM][JDBC Driver] CLI0600E Invalid connection handle or connection is closed. SQLSTATE=S1000.
Done
Internet

Figure 6-1 WebSphere started without sourcing the DB2 environment

- To check if the TMTP Management Server is up and running, type the following URL into your browser (this will only work for a nonsecure installation; for a secure installation, you will need to use the port 9446 and will need to import the appropriate certificates into your browser key store; this process is described below):

  `http://managementservername:9081/tmtp/servlet/PingServlet`

- If you use the secure installation of the TMTP Server, you can use the following procedure to check your SSL setup.

  Import the appropriate certificate into your browser key store.

  If you are checking to see if SnF should be able to connect to Management Server, the following is required.

  - Open the Store and Forward machines.kdb file using the IBM Key Management utility, that is, the key management tool, which can open kdb files.
  - Export the self signed personal certificate of the SnF machine to a PKCS12 format file (this is a format that the browser will be able to import). The resulting file should have a.p12 file extension.

– The export will ask if you want to use strong or weak encryption. Select weak encryption, as your browser will only be able to work with weak encryption.

  Now open your browser and select **Tools** → **Options** → **Content**. (we have only tried this with Internet Explorer version 6.x).

– Press the **Certificates** button. Import the exported.p12 file into the personal certificates of the browser.

– Now the following URL will tell you if SSL works between your machine and the Management Server using the certificate you imported above:

```
https://managementservername:9446/tmtp/servlet/PingServlet
```

If the Management Server works properly, you should see the statistics window shown in Figure 6-2 in your browser.

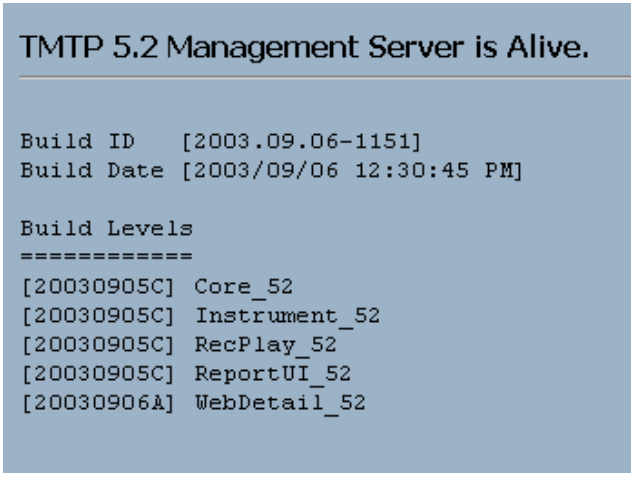


Figure 6-2 Management Server ping output

- To restart the TMTP server, log on to the WebSphere Application Server Administrative Console:

  ```
  http://WebSphere_server_hostname:9090/admin
  ```

  Go to the **Applications** → **Enterprise Applications** menu on the right side of the window; you can see the TMTPv5_2 application. Select the check box next to it and press **Stop** and then the **Start** button on the top of the panel.

- To stop and start the Store and Forward agent you have to restart the following services:

  – IBM Caching Proxy

  – Tivoli TransPerf Service

- To stop and start the Management Agent, you have to restart the following service:
  - Tivoli TransPerf Service

  Tip: Stopping the Management Agent will generally stop all of the associated behavior services; however, in the case of the QoS, we found that stopping the Management Agent would sometimes not stop the QoS service. If the QoS service does not stop, you will have to stop it manually.

- To redirect a Management Agent to another Store and Forward agent or directly to the Management Server, these steps need to be followed:
  - Open the *[MA_installation_folder]\config\endpoint.properties* file.
  - Change the *endpoint.msurl=https\://servername\:443* option to the new Store and Forward or Management Server host name.
  - Restart the Management Agent service.

  Important: The Management Agent can not be redirected to a different Management Server without reinstallation.

- To redirect a Store and Forward agent from one Store and Forward agent or to the Management Server directly, follow these steps:
  - Open the *[SnF_installation_folder]\config\snf.properties* file.
  - Edit the *proxy.proxy=https\://ibmtiv4.itsc.austin.ibm.com\:9446/tmtp/** option for the new Store and Forward or Management Server host name.
  - Restart the Store and Forward agent service.
- The following parameters are listed in the endpoint.properties file; however, changing them here will not affect the Management Agents behavior.
  - endpoint.uuid
  - endpoint.name
  - windows.password
  - endpoint.port
  - windows.user
- You can modify the location of the JKS files by editing the endpoint.keystore parameter in the endpoint.properties file and restarting the relevant service(s).
- Component management

  It is important to manage the data accumulated by TMTP. By default, data greater than 30 days old is cleared out automatically. This period can be

changed by selecting **Systems Administration** → **Components Management**. If your business requires longer-lasting historical data, you should utilize Tivoli Data Warehouse.

- Monitoring of TMTP system events:

  The following system events generated by TMTP are important TMTP status indicators and should be managed carefully by the TMTP administrator.

  - TEC-Event-Lost-Data
  - J2EE Arm not run
  - Monitoring Engine Lost ARM Connection
  - Playback Schedule Overrun
  - Policy Execution Failed
  - Policy Did Not Start
  - Policy Did Not Start
  - Management-Agent-Out-of-Service
  - TMTP BDH data transfer failed

  Generally, the best way to manage these events is for the event to be forwarded to the Tivoli Enterprise Console; however, other alternatives include generating an SNMP trap, sending an e-mail, or running a script. Event responses can be configured by selecting **Systems Administration** → **Configure System Event Details**.

6.1.1 Checking MBeans

The following procedure shows how to enable the HTTP Adapter for the MBean server on the Management Agent. This HTTP adapter is useful for troubleshooting purposes; however, it creates a security hole, so it should not be left enabled in a production environment. The TMTP installation disables this access by default.

The MBean server configuration file is named tmtp-sc.xml and is located in the $MA_HOME\config directory ($MA_HOME is the Management Agent home directory; by default, this is C:\Program Files\IBM\Tivoli\MA on a Windows machine). To enable the HTTP adaptor, you will need to add the section shown in Example 6-1 on page 183 to the tmtp-sc.xml file, and then restart the Tivoli transperf service/daemon.

Example 6-1 MbeanServer HTTP enable

```
<mbean
     class="com.ibm.tivoli.transperf.core.services.sm.HTTPAdapterService"
         name="TMTP:type=HTTPAdapter">
     <attribute name="Port"
         type="int" value="6969"/>
</mbean>
```

To access the MBean HTTP adapter, point your Web browser to `http://hostname:6969`. From the HTTP Adapter, you can control the MBean server as well as see any attributes of the MBean server. Using this interface is, of course, not supported; however, if you are interested in delving deeper into how TMTP works or troubleshooting some aspects of TMTP, it is useful to know how to set this access up. Figure 6-3 shows what will be displayed in your browser after successfully connecting to the MBean Servers HTTP adapter.

Figure 6-3 MBean Server HTTP Adapter

Some of the functions that can be performed from this interface are:

- List all of the MBeans
- Modify logging levels
- Show/change attributes of MBeans

- View the exact build level of each component installed on a Management Agent or the Management Server
- Stop and start the ARM agent without stopping and starting the Tivoli TransPerf service/daemon
- Change upload intervals (from the Management Server)

6.2 Configuring the ARM Agent

The ARM engine uses a configuration file to control how it runs, the amount of system resources it uses, and so on. The name of this file is tapm_ep.cfg. This file is created on the Management Agent the first time the ARM engine is run. The location of this file is one of the following:

Windows	$MA_DIR\arm\apf\tapm_ep.cfg
UNIX	$MA_DIR/arm/apf/tapm_ep.cfg

Where $MA_DIR is the root directory where the TMTP Version 5.2 agent is installed.

The contents of this file are read when the ARM engine starts. In general, you will not have to change the values in this file, as the defaults will cover most environments. If changes are made to this file, they are not loaded until the next time the ARM engine is started.

Note: The ARM agent (tapmagent.exe) is started by the Management Agent, that is, to start and stop the ARM agent, you will need to stop and start the Tivoli Management Agent. On Windows-based platforms, this is achieved by stopping and starting the “Tivoli TransPerf Service” (jmxservice.exe). On UNIX platforms, the Management Agent is stopped and started using the stop_tmtpd.sh and start_tmtpd.sh scripts.

The contents of the file are organized in stanzas (denoted by a [character followed by the section name and ending with a] character). Within each section are a number of key=value pairs.

Some of the more interesting keys are described below.

The entry:

```
[ENGINE::LOG]
   LogLevel=1
```

defines the level of logging that the ARM engine will use. The valid values for this key are shown in Table 6-1 on page 185.

Table 6-1 ARM engine log levels

Value	Description
1	Minimum logging. Error conditions and some performance logging.
2	Medium logging. All of 1 and more.
3	High logging. All of 2 and much more.

The logging from the Management Agent ARM engine is, by default, sent to one of the following files:

Windows C:\Program Files\ibm\tivoli\common\BWM\logs\tapmagent.log
UNIX /usr/ibm/tivoli/common/BWM/logs/tapmagent.log

If you are experiencing problems with the ARM agent, you can set this key to 3 and stop and start the Management Agent to get level 3 logging.

These two keys:

```
[ENGINE::INTERNALS]
   IPCAppToEngSize=500
   IPCEngToAppSize=500
```

define the size of internal buffers used for communications between ARM instrumented applications and the ARM engine. The IPCAppToEngSize key defines the number of elements used for ARM instrumented applications to communicate to the ARM engine. Likewise, the IPCEngToAppSize key defines the number of elements used for communications from the ARM engine back to the ARM instrumented applications.

In this example, 500 elements are assigned to each of these buffers. The larger these buffers are, the more memory is taken up by the ARM engine. If the application being monitored is a single threaded application, and only one application is being monitored, then these numbers can be decreased. This is not normally the case. Most applications are multithreaded and need a large number of entries here. If the number of entries is set too low, applications making many calls to the ARM engine will be blocked by the ARM engine until an unused entry is found that will slow the ARM instrumented application.

In general, changes to these two entries should only be necessary on a UNIX Management Agent and the values for the two entries should be kept the same.

If the ARM engine will not start and the log file shows errors in IPC, attempt to lower these values.

Some other interesting key value pairs include:

TransactionIDCacheSize=100000
: This is the number of transactions that are allowed to be active at any specific point in time. Once this limit is reached, the least recently run transaction mapping is removed from memory and a arm_getid call must proceed any future start calls for that transaction ID mapping.

TransactionIDCacheRemoveCount=10
: This is the number of transactions we flush from the cache when the above limit is reached.

PolicyCacheSize=100000
: This is the number of transaction IDs to policy mappings kept in memory at any one time. This saves TMTP from having to perform regular expression matches for every policy each time it sees a transaction. Making this larger than TransactionIDCacheSize really does not have any value, but setting it equal is a good idea. This cache has to be flushed completely every time a management policy is added to the agent.

PolicyCacheRemoveCount=10
: When the above cache size limit is reached, this many entries are removed.

EdgeCacheSize=100000
: This is the number of unique edges TMTP has "seen" that are kept in memory to avoid sending duplicate new edge notifications to the Management Server. This cache can be lowered or raised freely depending on your memory consumption desired. Lowering it can potentially cause more network agent and Management Server load, but less memory requirements on the agent.

EdgeCacheRemoveCount=10
: This is the number of edge entries to remove when the above limit is reached.

MaxAggregators=1000000
: This is the maximum number of unique aggregators to keep in memory for any one hour period. It is advisable to have this set as high as possible, given your memory limit desires for the Management Agent. Warnings will be logged when this limit is reached and the old aggregator in memory will be flushed to disk.

ApplicationIDfile=applications.dat
: The file name to store previously seen applications.

RawTransactionQueueSize=500
: This is the maximum number of simultaneously started transactions that have not yet completed that TMTP will allow.

CompletedTransactionQueueSize=250

This is the maximum size of the completed transaction queue. These are transactions that have completed and are awaiting processing. When this limit is reached, the ARM STOP call will block while it waits for transactions to be processed and space to be freed. This can be raised at the expense of memory to allow your system to handle large rapid bursts of transactions to occur without noticeable slowdown of the response time.

Most of the other Key/Value pairs in this file are legacy and do not have any effect on the behavior of the agent.

ARM Engine log file

As described above, the Management Agent ARM engine, by default, sends all trace logs to one of the following files:

Windows	C:\Program Files\ibm\tivoli\common\BWM\logs\tapmagent.log
UNIX	/usr/ibm/tivoli/common/BWM/logs/tapmagent.log

The location of this file is determined by the file.fileName entry in one of the following files:

Windows	$MA_DIR\config\tapmagent-logging.properties
UNIX	$MA_DIR/config/tapmagent-logging.properties

To change the location of the ARM engine trace log file, simply change the file.fileName entry in this file. Please note that the logging levels specified in this file have no effect. To change logging levels for the ARM agent, you will need to modify the logging level entries in the tmtp-sc.xml file, as described in the previous section.

To get a more condensed version of the ARM engine trace log, set the fmt.className entry to ccg_basicformatter (this line exists in the tapmagent-logging.properties file and only needs to be uncommented; comment out the existing fmt.className line).

ARM data

The ARM Engine stores the data that it collects in the following directory in a binary format prior to being uploaded to the Management Server:

```
$MA_HOME\arm\mar\.Dat
```

By default, this directory is hidden. At each the end of each upload period, this data is consolidated and placed into the $MA_HOME\arm\mar\.Dat\update

directory, from where it is picked up by the Bulk Data Transfer service to be forwarded to the Management Server.

If instance records are being collected by the ARM agent another directory called $MA_HOME\arm\mar\.Dat\current will be automatically created, which will contain subdirectories for each of the instance records.

6.3 J2EE monitoring maintenance

During our work on this redbook, we ran into a small number of problems using the J2EE monitoring component. Most of these issues were because we were using prerelease code for much of our work. While troubleshooting these issues, the following steps were useful and may prove useful in a production environment.

ARM records not created

If you are not receiving ARM records, you can use the following steps to ensure that there are no problems with the policy, J2EE, or ARM. These steps will verify that the ARM engine recognizes the policy and that ARM records are being generated by J2EE.

- Verify that the J2EE component successfully installed.

  Verify in the User Interface "Work with Agents" section that the J2EE component says `RUNNING`.

  Possible problem:

  UI does not say `RUNNING…` .

  Possible solution:

  If the UI says `INSTALL_IN_PROGRESS`, then keep waiting. If you wait for an extremely long time (30 minutes), and you checked **Automatically restart Application server**, then the install is hung. You will need to manually stop and restart the application server on the Management Agent. If you do this and it does not switch to RUNNING, open a defect on Instrument.

  If the UI says `INSTALL_RESTART_APPSERVER`, then restart the appserver on the Management Agent and rerun the PetStore or other application to collect ARM data.

  If the UI says `INSTALL_FAILED`, then verify that you entered the correct info for your J2EE component. If you think everything was entered correctly, then open a defect on Instrument.

- Verify that the J2EE appserver is instrumented.

  Verify that the following files/directory structure exists:

  – Management Agent

  – Common J2EE Behavior files

  – <MA_HOME>/app/instrument/appServers/<UUID>/BWM/logs/trace.log

  Possible problem:

  If this file does not exist, then the application server has not been instrumented or the application server needs to be restarted for the instrumentation to take affect.

  Possible solution:

  Restart the appserver and access one of your instrumented applications (that is, an application that you have defined J2EE a policy for). If the trace log still does not exist, then verify you entered the correct information into the policy. If you have entered the correct information and the trace file has not been created, then you may have encountered a defect, in which case you will need to log a PMR with IBM Tivoli Support.

- Verify that your Listening Policy exists on Management Agent.

  This step will verify that the Management Server sent the Management Agent your listening policy correctly; in order for this section to work, you will need to re-enable access to the HTTP Adaptor of the MBeanServer on your Management Agent. The procedure to do this is described in 6.1.1, “Checking MBeans” on page 182.

  Open a browser and go to the address `http://MAHost:6969`, where MAHost is the host name of the Management Agent you wish to check.

  a. Select **Find an MBean**.

  b. Select **Submit Query**.

  c. Select **TMTP:type=MAPolicyManager**.

  Verify that your policy is listed here (the URI pattern you have specified in the policy will be listed).

  Possible problem:

  If the policy does not exist, but you selected “Send to Agents Now” in your policy, then there was a problem sending the policy from the Management Server to the Management Agent.

  Possible solution:

  To get the policy:

  a. Select **pingManagementServer()**.

b. Select **Invoke Operation**.

Click **Back** twice and then press F5 to refresh the screen.

Verify that your policy is listed here. If this has not fixed your problem, you may have encountered a defect and should open a PMR with IBM Tivoli Support.

- Verify that ARM is receiving transactions.

  This step will verify that ARM is using your listening policy correctly and that J2EE is submitting ARM requests.

  Open the ARM engine log file in which is located in the Tivoli Common Directory. On Windows, it is located in C:\Program Files\ibm\tivoli\common\BWM\logs\tapmagent.log.

  Search this file for arm_start. If it exists, then J2EE is correctly instrumented and making ARM calls.

  Possible problem:

  If arm_start does not exist, then J2EE could be instrumented incorrectly. Verify in the UI that the J2EE component says `RUNNING`.

  Possible solution:

  If there is no arm_start but the UI says `RUNNING`, you may have encountered a defect and should open a PMR with IBM Tivoli Support.

  If arm_start exists, then search the file for WriteNewEdge. If this exists, then ARM has successfully matched a J2EE edge with an existing policy.

  Possible problem:

  If arm_start exists but WriteNewEdge does not exist, then there could be a problem with your listening policy or your have not run an instrumented application.

  At this point, also check to see if ARM_IGNORE_ID exists. If it does, then the edge URI for the listening policy is not matching the edge that J2EE is sending.

  Possible solution:

  Verify that you have run an application that would match your policy. Verify that the listening policy is on the Management Agent and that the URI pattern matches what the URI you are clicking on for the application on the Management Agent's appserver. If this is still a problem, then you may have to open a PMR with IBM Tivoli Support.

6.4 TMTP TDW maintenance tips

This section provides information about maintaining and troubleshooting the Tivoli Data Warehouse.

Backing up and restoring

The dbrest.bat script in the misc\tools directory is an example script that shows you how to restore the three databases on an NT or 2000 Microsoft® Windows System.

Pruning

If you have established a schedule to automatically run the data mart ETL process steps on a periodic basis, occasionally manually prune the logs in the directory %DB2DIR%\logging.

The BWM_m05_s050_mart_prune step prunes the hourly, daily, weekly, and monthly fact tables as soon as they have data older than three months.

If you schedule the data mart ETL process to run daily, as recommended, you do not need to schedule pruning separately.

Duplicate row problem due to Source ETL process hangs

Problem:

The TMTP Version 5.2 process BWM_c10_cdw_process hangs and you restart the Data Warehouse or DB2. When you then try to rerun the BWM_c10_cdw_process, you will get duplicate row problem (see Figure 6-4 on page 192). This is because the TDW keeps a pointer to the last record it has processed. If the TDW is restarted during processing, the pointer will be incorrect and the BWM_c10_cdw_process may re-process some data.

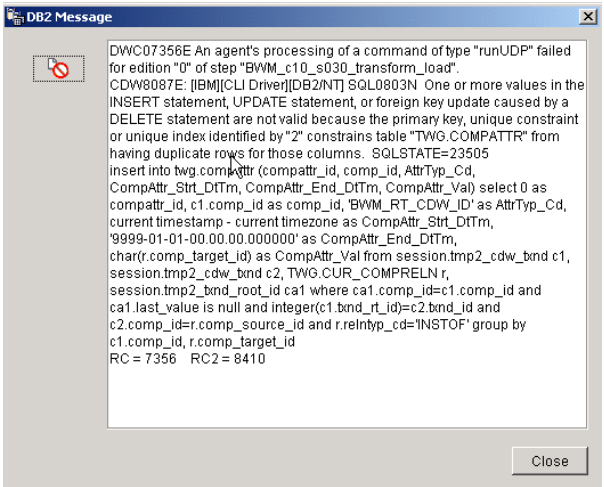

DB2 Message
DWC07356E An agent's processing of a command of type "runUDP" failed for edition "0" of step "BWM_c10_s030_transform_load".
CDW8087E: [IBM][CLI Driver][DB2/NT] SQL0803N One or more values in the INSERT statement, UPDATE statement, or foreign key update caused by a DELETE statement are not valid because the primary key, unique constraint or unique index identified by "2" constrains table "TWG.COMPATTR" from having duplicate rows for those columns. SQLSTATE=23505
insert into twg.compattr (compattr_id, comp_id, AttrTyp_Cd, CompAttr_Strt_DtTm, CompAttr_End_DtTm, CompAttr_Val) select 0 as compattr_id, c1.comp_id as comp_id, 'BWM_RT_CDW_ID' as AttrTyp_Cd, current timestamp - current timezone as CompAttr_Strt_DtTm, '9999-01-01-00.00.00.000000' as CompAttr_End_DtTm, char(r.comp_target_id) as CompAttr_Val from session.tmp2_cdw_txnd c1, session.tmp2_cdw_txnd c2, TWG.CUR_COMPRELN r, session.tmp2_txnd_root_id ca1 where ca1.comp_id=c1.comp_id and ca1.last_value is null and integer(c1.txnd_rt_id)=c2.txnd_id and c2.comp_id=r.comp_source_id and r.relntyp_cd='INSTOF' group by c1.comp_id, r.comp_target_id
RC = 7356 RC2 = 8410
Close

Figure 6-4 Duplicate row at the TWH_CDW

Solution:

The cleancdw.sql script (see Example 6-2) will clean the BWM source information if we need to clean TMTP database information from TWH_CDW.

Example 6-2 cleancdw.sql

```
CONNECT to twh_cdw
Delete from TWG.compattr
Delete from TWG.compreln
Delete from TWG.msmt
Delete from TWG.comp
Delete from bwm.comp_name_long
Delete from bwm.comp_attr_long
UPDATE TWG.Extract_control SET EXTCTL_FROM_INTSEQ=-1
UPDATE TWG.Extract_control SET EXTCTL_TO_INTSEQ=-1
```

We then need to run the resetsequences.sql script (see Example 6-3) to reset the TMTP ETL1 process after running the cleancdw.sql script.

Example 6-3 resetsequences.sql

```
CONNECT to twh_cdw
UPDATE TWG.Extract_control SET EXTCTL_FROM_INTSEQ=-1
UPDATE TWG.Extract_control SET EXTCTL_TO_INTSEQ=-1
UPDATE TWG.Extract_control SET ExtCtl_From_DtTm='1970-01-01-00.00.00.000000'
UPDATE TWG.Extract_control SET ExtCtl_To_DtTm='1970-01-01-00.00.00.000000'
```

Tools

The extract_win.bat script resets the Extract Control window for the warehouse pack. You should use this script only to restart the Extract Control window for the BWM_m05_Mart_Process. If you want to reset the window to the last extract, use the extract_log to get the last values of each DB2 (BWM) extract.

The bwm_c10_CDW_process.bat script executes the BWM_c10_CDW_Process from the command line. The bwm_m05_MART_Process.bat script executes the BWM_m05_Mart_Process from the command line.

The bwm_upgrade_clear.sql script undoes all the changes that the bwm_c05_s030_upgrade_convertdata process made. This script helps with troubleshooting for the IBM Tivoli Monitoring for Transaction Performance Version 5.1 upgrade process. If errors are raised during the data converting, use this script to help clear up the converted data. After the problem is fixed, you can rerun the bwm_c05_s030_upgrade_convertdata process to continue the upgrade and migration.

For more details about managing the Tivoli Data Warehouse, see the Tivoli Enterprise Data Warehouse manuals and the following Redbooks:

- *Planning a Tivoli Enterprise Data Warehouse Project*, SG24-6608
- *Introduction to Tivoli Enterprise Data Warehouse*, SG24-6607

6.5 Uninstalling the TMTP Management Server

De-installing TMTP is generally straightforward and well covered in the TMTP manuals. Uninstallation on the UNIX/Linux platform is a little more problematic, so we have included some information below to make this easier.

6.5.1 The right way to uninstall on UNIX

The following steps are required to uninstall TMTP after completing a typical install (that is, an embedded install). The uninstall program for the TMTP Management Server will not uninstall the WebSphere and DB2 installations that were installed by the embedded install, that is, they will have to be performed using their own native uninstallation procedures.

1. Uninstall the TMTP Management Server by running the following command:

   ```
   $MS_HOME/_uninst52/uninstall.bin
   ```

2. Uninstall WebSphere by running the following commands (by default, WebSphere is installed in a subdirectory of the Management Server home directory by the embedded install process):

   ```
   $MS_HOME/WAS/bin/stopServer.sh server1 -user userid -password password
   $MS_HOME/WAS/_uninst/uninstall
   ```

3. Uninstall DB2:

   a. Source the DB2 profile; this will set the appropriate environment variables.

      ```
      . $INSTDIR/sqllib/db2profile
      ```

      $INSTDIR is the db2 instance home directory.

   b. Drop the administrative instance.

      ```
      $DB2DIR/instance/dasdrop
      ```

   c. List the db2 instances.

      ```
      $DB2DIR/bin/db2ilist
      ```

   d. For each instance listed above, run:

      ```
      $DB2DIR/instance/db2idrop <instance>
      ```

   e. From the DB2 install directory, run the db2 deinstall script:

      ```
      db2_deinstall
      ```

   f. Remove the DB2 admin, instance, and fence users, and delete their home directories. On many UNIX platforms, you can delete users with the following command:

      ```
      userdel -r <login name> # -r removes home directory
      ```

      This should remove entries from /etc/passwd and /etc/shadow.

   g. Remove /var/db2 if no other version of DB2 is installed.

   h. Delete any DB2-related lines from /etc/services.

   i. On Solaris, check the size of textfile /var/adm/messages; DB2 can sometimes increase it to hundreds of megabytes. Truncate this file if required.

   j. Remove any old db2 related files in /tmp (there will be some log files and other nonessential files here).

6.5.2 The wrong way to uninstall on UNIX

Experienced UNIX administrators are often tempted to uninstall using a brute force method, that is, deleting the directories associated with the installs. This will work, but you should keep the following points in mind:

- The DB2 installation will create several new users (generally, db2inst1, db2fenc1, and so on), which will need to be deleted (see the procedure for removing DB2 above).
- IBM Tivoli keeps a record of each product it has installed in a file named vpd.properties. This file is located in the home directory of the user used for the installation (in our case, /root). If this file is not modified, it will prevent later reinstall attempts for TMTP, as it may indicate to the installation process that a particular product is already installed. Generally, you will only need to remove entries in this file that relate to products you have manually deleted. In our test environment, it was generally safe to delete the file, as the only IBM Tivoli product we had installed was TMTP.
- On UNIX platforms, WebSphere Application Server and DB2 will generally use native package install processes, for example, RPM on Linux. This means that a brute force install may leave the package manager information in an inconsistent state.

6.5.3 Removing GenWin from a Management Agent

Chapter 6, “Removing a Component” of the *IBM Tivoli Monitoring for Transaction Performance Installation Guide Version 5.2.0*, SC32-1385 covers uninstalling the GenWin behavior from a Management Agent. One of the points it highlights is that you must delete the Rational Robot project that you are using for the GenWin behavior prior to removing the GenWin behavior. This point is important, as removing the GenWin behavior will delete the directory used by the Rational Robot project associated with that GenWin behavior. The ramification of this is that if you have not previously deleted the Rational Robot project, you will not be able to create a new Rational Robot project with the same name as this project (you will get the error message shown in Figure 6-5 on page 196), that is, you end up with an orphan project that is not displayed in the Rational Administrator tool, and the name of which cannot be reused.

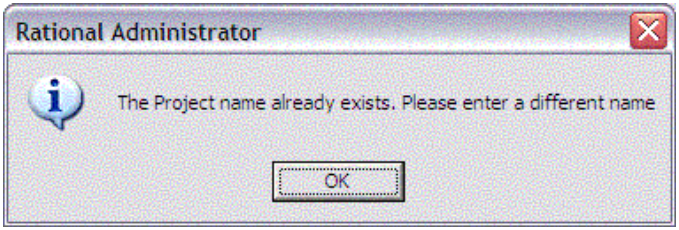


Figure 6-5 Rational Project exists error message

If you find yourself in this unfortunate position, the following procedure may help. The Rational Administrator maintains its project list under the following registry key:

```
HKEY_CURRENT_USER\Software\Rational Software\Rational Administrator\ProjectList
```

If you delete the “orphan” project name from this key, you should now be able to reuse it.

6.5.4 Removing the J2EE component manually

In most instances, you should use the Management Server interface to remove the J2EE component from a Management Agent. Doing this will remove the J2EE instrumentation from the Web Application Server correctly. Occasionally, you may find yourself in a situation where the Management Agent is unable to communicate with the Management Server when you need to remove the J2EE component. The best way of removing the J2EE component in this situation is to just uninstall the Management Agent, as this will also remove the J2EE instrumentation from your Web Application Server. Very occasionally, you may get yourself into the position where you need to remove the J2EE instrumentation from the Web Application Server manually. If this happens, you can use the following procedure as a last resort.

Important: You should only use this procedure when all else fails.

Manual J2EE uninstall on WebSphere 4.0

1. Start the WebSphere 4 Advanced Administrative Console on the computer on which the instrumented application server resides. Expand the “WebSphere Administrative Domain” tree on the left and select the application server that has been instrumented (see Figure 6-6 on page 197).

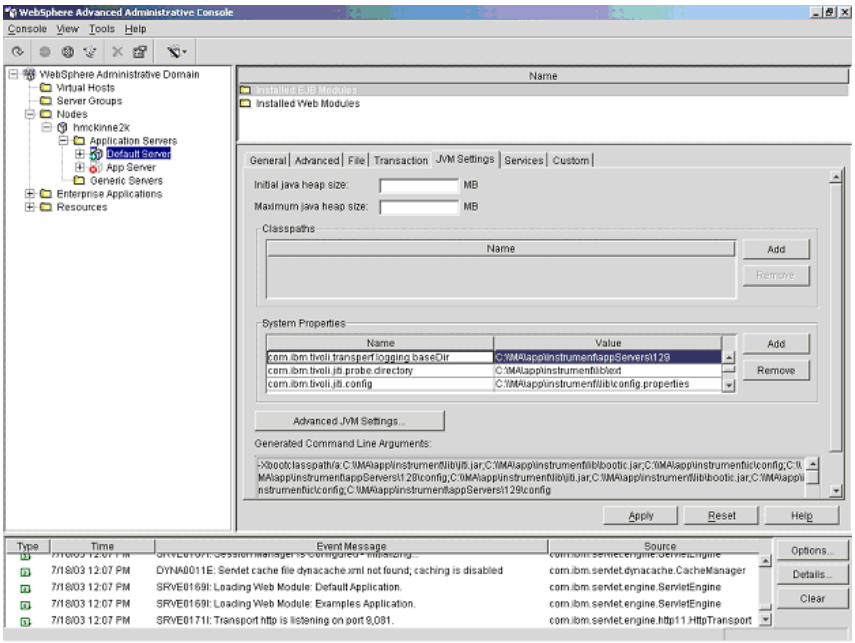


Figure 6-6 WebSphere 4 Admin Console

2. On the right panel, select the tab labeled **JVM Settings**. Under the System Properties table, remove each of the following eight properties:
   - jlog.propertyFileDir
   - com.ibm.tivoli.transperf.logging.baseDir
   - com.ibm.tivoli.jiti.probe.directory
   - com.ibm.tivoli.jiti.config
   - com.ibm.tivoli.jiti.logging.FileLoggingImpl.logFileName
   - com.ibm.tivoli.jiti.registry.Registry.serializedFileName
   - com.ibm.tivoli.jiti.logging.IloggingImpl
   - com.ibm.tivoli.jiti.logging.NativeFileLoggingImpl.logFileName
3. Click the **Advanced JMV Settings…** button, which opens the Advanced JVM Settings window. In the Command line arguments text box, remove the entry `-Xrunijitipi:<MA>\app\instrument\lib\jiti.properties`. In the Boot classpath (append) text box, remove the following entries:
   - <MA>\app\instrument\lib\jiti.jar, <MA>\app\instrument\lib\bootic.jar
   - <MA>\app\instrument\ic\config
   - <MA>\app\instrument\appServers\<n>\config

- <MA>\app\instrument\lib\jiti.jar
- <MA>\app\instrument\lib\bootic.jar
- <MA>\app\instrument\ic\config
- <MA>\app\instrument\appServers\<n>\config

  where <MA> represents the root directory where the TMTP Version 5.2 Management Agent has been installed, and <n> will be a random number.

4. Click the **OK** button, which will close the Advanced JVM Settings window.
5. Back in the main WebSphere Advanced Administrative Console window, click the **Apply** button.
6. The administrative node on which the instrumented application server is installed must be shut down so that the TMTP files that have been installed under the WebSphere Application Server directory may be removed. On the WebSphere Administrative Domain tree on the left, select the node on which the instrumented application server is installed. Right-click on the node, and select **Stop**.

   Warning: This will stop all application servers running on that node.

7. After the administrative node is stopped, remove the following nine files from the directory <WAS_HOME>\AppServer\lib\ext, where <WAS_HOME> is the home directory where WebSphere Application Server Advanced Edition is installed:
   - armjni.jar
   - copyright. jar
   - core_util.jar
   - ejflt.jar
   - eppam.jar
   - jffdc.jar
   - jflt.jar
   - jlog.jar
   - probes.jar
8. Remove the file <WAS_HOME>\AppServer\bin\ijitipi.dll.
9. The administrative node and application server may now be restarted.

Manual J2EE uninstall on WebSphere 5.0

1. Start the WebSphere 5 Application Server Administrative Console on the computer on which the instrumented application server resides, or on the Network Deployment server.
2. In the navigation tree on the left, expand **Servers**. Click on the **Application Servers** link.
3. In the Application Servers table on the right, click on the application server that has been instrumented.
4. Under the Additional Properties table, click the **Process Definition** link.
5. Under the Additional Properties table, click the **Java Virtual Machine** link.
6. Under the General Properties table, look for the **Generic JVM Argument** field (see Figure 6-7).

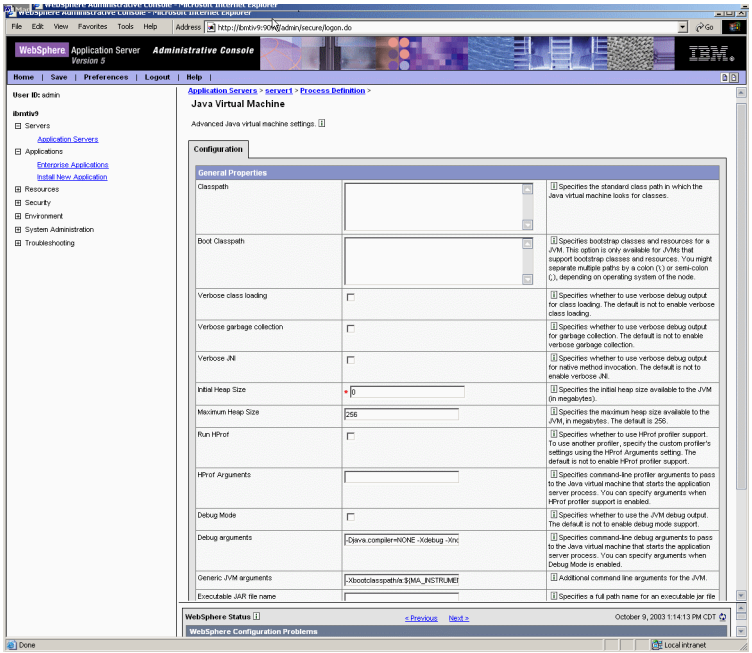


Figure 6-7 Removing the JVM Generic Arguments

7. Remove all of the following entries from this field:
   - Xbootclasspath/a:${MA_INSTRUMENT}\lib\jiti.jar; ${MA_INSTRUMENT}\lib\bootic.jar; ${MA_INSTRUMENT}\ic\config; ${MA_INSTRUMENT_APPSERVER_CONFIG}
   - Xrunijitipi:${MA_INSTRUMENT}\lib\jiti.properties

- Dcom.ibm.tivoli.jiti.config=${MA_INSTRUMENT}\lib\config.properties
- Dcom.ibm.tivoli.transperf.logging.baseDir=${MA_INSTRUMENT}\appServers\130
- Dcom.ibm.tivoli.jiti.logging.ILoggingImpl = com.ibm.tivoli.transperf.instr.controller.TMTPConsoleLoggingImpl
- Dcom.ibm.tivoli.jiti.logging.FileLoggingImpl.logFileName = ${MA_INSTRUMENT}\BWM\logs\jiti.log
- Dcom.ibm.tivoli.jiti.logging.NativeFileLoggingImpl.logFileName = ${MA_INSTRUMENT}\BWM\logs\native.log
- Dcom.ibm.tivoli.jiti.probe.directory=E:\MA\app\instrument\appServers\lib
- Dcom.ibm.tivoli.jiti.registry.Registry.serializedFileName = ${MA_INSTRUMENT}\lib\registry.ser
- Djlog.propertyFileDir = ${MA_INSTRUMENT_APPSERVER_CONFIG}
- Dws.ext.dirs = E:\MA\app\instrument\appServers\lib

8. Click the **OK** button.
9. Click the **Save Configuration** link at the top of the page.
10. Click the **Save** button on the new page that appears.
11. In order to remove TMTP files that have been installed under the WebSphere Application Server directory, all application servers running on this node must be shutdown. Stop each application server with the **`stopServer`** command.
12. After each application server has been stopped, remove the following nine files from the directory <WAS_HOME>\AppServer\lib\ext, where <WAS_HOME> is the home directory where WebSphere Application Server is installed:
    - armjni.jar
    - copyright. jar
    - core_util.jar
    - ejflt.jar
    - eppam.jar
    - jffdc.jar
    - jflt.jar
    - jlog.jar
    - probes.jar
13. Remove the file <WAS_HOME>\AppServer\bin\ijitipi.dll.
14. The application servers running on this node may now be started.

Manual uninstall of J2EE component on Weblogic 7

The following procedure outlines the steps needed to perform a manual uninstall of the TMTP J2EE component from a Weblogic server.

1. The WebLogic 7 installation has two options: “A script starts this server” and “Node Manager Starts this server”. One or both of those options can be selected when J2EE Instrumentation is installed. If J2EE Instrumentation was installed with “A script starts this server”, follow steps 2 and 3. If the J2EE Instrumentation used “Node Manager starts this server”, follow steps 4 through 7. Finally, follow steps 8-10 to clean up any files that were used by J2EE Instrumentation.
2. Edit the script that starts the WebLogic 7 server. The script is a parameter to the installation, which may be something similar to C:\beaHome701\user_projectsAJL\mydomain\startPetStore.cmd.
3. In the script, remove the lines from @rem Begin TMTP AppIDnnn to @rem End TMTP AppIDnnn, where nnn is a UUID, such as 101, 102, and so on. The text to be removed will be similar to Example 6-4.

Example 6-4 Weblogic TMTP script entry

```
@rem Begin TMTP AppID169
if "%SERVER_NAME%"=="thinkAndy" set
PATH=C:\\ma.2003.07.03.0015\app\instrument\\lib\windows;%PATH%

if "%SERVER_NAME%"=="thinkAndy" set MA=C:\\ma.2003.07.03.0015

if "%SERVER_NAME%"=="thinkAndy" set MA_INSTRUMENT=%MA%\app\instrument

if "%SERVER_NAME%"=="thinkAndy" set
JITI_OPTIONS=-Xbootclasspath/a:%MA_INSTRUMENT%\lib\jiti.jar;%MA_INSTRUMENT%\lib
\bootic.jar;%MA_INSTRUMENT%\ic\config;%MA_INSTRUMENT%\appServers\169\config
-Xrunjitipi:%MA_INSTRUMENT%\lib\jiti.properties
-Dcom.ibm.tivoli.jiti.config=%MA_INSTRUMENT%\\lib\config.properties
-Dcom.ibm.tivoli.transperf.logging.baseDir=%MA_INSTRUMENT%\appServers\169
-Dcom.ibm.tivoli.jiti.logging.ILoggingImpl=com.ibm.tivoli.transperf.instr.contr
oller.TMTPConsoleLoggingImpl
-Dcom.ibm.tivoli.jiti.logging.FileLoggingImpl.logFileName=%MA_INSTRUMENT%\BWM\l
ogs\jiti.log
-Dcom.ibm.tivoli.jiti.logging.NativeFileLoggingImpl.logFileName=%MA_INSTRUMENT%
\BWM\logs\native.log
-Dcom.ibm.tivoli.jiti.registry.Registry.serializedFileName=%MA_INSTRUMENT%\lib\
WLRegistry.ser -Djlog.propertyFileDir=%MA_INSTRUMENT%\appServers\169\config

if "%SERVER_NAME%"=="thinkAndy" set JAVA_OPTIONS=%JITI_OPTIONS% %JAVA_OPTIONS%

if "%SERVER_NAME%"=="thinkAndy" set
CLASSPATH=%CLASSPATH%;C:\beaHome701\weblogic700\server\lib\ext\probes.jar;C:\be
```

```
aHome701\weblogic700\server\lib\ext\ejflt.jar;C:\beaHome701\weblogic700\server\
lib\ext\jflt.jar;C:\beaHome701\weblogic700\server\lib\ext\jffdc.jar;C:\beaHome7
01\weblogic700\server\lib\ext\jlog.jar;C:\beaHome701\weblogic700\server\lib\ext
\copyright.jar;C:\beaHome701\weblogic700\server\lib\ext\core_util.jar;C:\beaHom
e701\weblogic700\server\lib\ext\armjni.jar;C:\beaHome701\weblogic700\server\lib
\ext\eppam.jar
@rem End TMTP AppID169
```

4. Point a Web browser to the WebLogic Server Console. The address will be something similar to `http://myHostname.com:7001/console`.
5. In the left hand applet frame, select the domain and server that was configured with J2EE Instrumentation. Click on the **Remote Start** tab of the configuration for the server (see Figure 6-8).

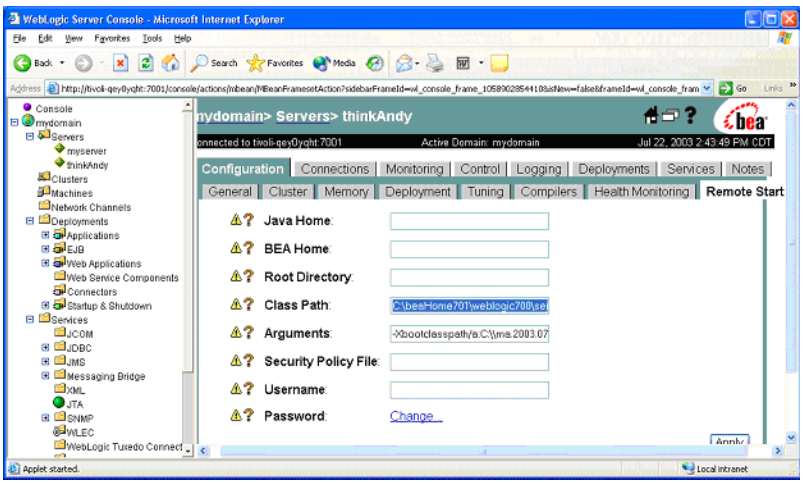


Figure 6-8 WebLogic class path and argument settings

6. Edit the Class Path and Arguments fields to restore them to the original value before deploying J2EE Instrumentation. If these two fields were blank before installing J2EE Instrumentation, then they should be reverted to being blank. If these two fields had configuration not related to J2EE Instrumentation, only remove the values that were added by J2EE Instrumentation. The values added by the J2EE Instrumentation install will be similar to those values shown in Example 6-5.

Example 6-5 Weblogic Class Path and Arguments fields

```
Class Path:
C:\beaHome701\weblogic700\server\lib\ext\probes.jar;C:\beaHome701\weblogic700\s
erver\lib\ext\ejflt.jar;C:\beaHome701\weblogic700\server\lib\ext\jflt.jar;C:\be
aHome701\weblogic700\server\lib\ext\jffdc.jar;C:\beaHome701\weblogic700\server\
```

```
lib\ext\jlog.jar;C:\beaHome701\weblogic700\server\lib\ext\copyright.jar;C:\beaH
ome701\weblogic700\server\lib\ext\core_util.jar;C:\beaHome701\weblogic700\serve
r\lib\ext\armjni.jar;C:\beaHome701\weblogic700\server\lib\ext\eppam.jar

Arguments:
-Xbootclasspath/a:C:\\ma.2003.07.03.0015\app\instrument\lib\jiti.jar;
C:\\ma.2003.07.03.0015\app\instrument\lib\bootic.jar;C:\\ma.2003.07.03.0015\app
\instrument\ic\config;C:\\ma.2003.07.03.0015\app\instrument\appServers\178\conf
ig -Xrunjitipi:C:\\ma.2003.07.03.0015\app\instrument\lib\jiti.properties
-Dcom.ibm.tivoli.jiti.config=C:\\ma.2003.07.03.0015\app\instrument\\lib\config.
properties
-Dcom.ibm.tivoli.transperf.logging.baseDir=C:\\ma.2003.07.03.0015\app\instrumen
t\appServers\178
-Dcom.ibm.tivoli.jiti.logging.ILoggingImpl=com.ibm.tivoli.transperf.instr.contr
oller.TMTPConsoleLoggingImpl
-Dcom.ibm.tivoli.jiti.logging.FileLoggingImpl.logFileName=C:\\ma.2003.07.03.001
5\app\instrument\BWM\logs\jiti.log
-Dcom.ibm.tivoli.jiti.logging.NativeFileLoggingImpl.logFileName=C:\\ma.2003.07.
03.0015\app\instrument\BWM\logs\native.log
-Dcom.ibm.tivoli.jiti.registry.Registry.serializedFileName=C:\\ma.2003.07.03.00
15\app\instrument\lib\WLRegistry.ser
-Djlog.propertyFileDir=C:\\ma.2003.07.03.0015\app\instrument\appServers\178\con
fig
```

7. Click **Apply** to apply the changes to the Class Path and Arguments fields.
8. Stop the WebLogic Application Server that was instrumented with J2EE Instrumentation.
9. After the application server has been stopped, remove the following nine files from the directory <WL7_HOME>\server\lib\ext, where <WL7_HOME> is the home directory of the WebLogic 7 Application Server:
   - armjni.jar
   - copyright.jar
   - core_util.jar
   - ejflt.jar
   - eppam.jar
   - jffdc.jar
   - jflt.jar
   - jlog.jar
   - probes.jar.

   After those nine files are removed, remove the empty <WL7_HOME>\server\lib\ext directory.

10. Remove the file <WL7_HOME>\server\bin\jitipi.dll or <WL7_HOME>\server\bin\ijitipi.dll file, if it exists. Some OS platforms use jitipi.dll and some OS platforms use ijitipi.dll.

> **Note:** The [i]jitipi.dll file may not exist in <WL7_HOME>\server\bin, depending on the version of J2EE Instrumentation. If it does not exist in this directory, it is in the Management Agent's directory, and can be left in the Management Agent's directory without any harm.

6.6 TMTP Version 5.2 best practices

This section describes our recommendations on how to implement and configure TMTP Version 5.2 to maximize effectiveness and performance in your production environment. Please note that although the following recommendations are general and suitable to most typical production environments, you may need to customize configurations for your environment and particular requirements.

Overview of recommendations

- Use the following default J2EE Monitoring settings for long term monitoring during normal operation in the production environment.
  - Only record aggregate records.
  - Discovery Policies for J2EE and QoS transactions should be run and then disabled once listening policies have been created off the discovered transactions.

    > **Note:** The Discovery Policies may be re-enabled at a future date if further transaction discovery is required.

  - Use a 20% sampling rate.
  - Set low tracing detail.
- Define the URI filters as narrow as possible to match the transaction patterns you are interested in monitoring. This will optimize monitoring overhead during normal operation in the production environment. The narrow URI filters also help the effectiveness of analysis of TMTP reports, as you can selectively investigate transaction data of interest.
- It is suggested to avoid using regular expressions that contain wildcard (.*) in the middle of URI filter, if possible.
- Only turn up the tracing details when a performance or availability violation is detected for the J2EE application server to allow for quick debugging of the

situation. It is recommended for high traffic Web sites to set the Sample Rate lower than 20% when a tracing detail higher than the “Low” level is used. Setting the maximum number of sample per minute instead of the sample rate is also recommended to better regulate monitoring overhead during a high traffic period.

- In a production environment, we recommend collecting Aggregate Data Only. TMTP will automatically collect a certain number of Instance records when a failure is detected. It is not recommended to collect Aggregate and Instance records during normal operation in a production environment, as it may generate overwhelming data.
- In a large-scale environment with more than 100 Management Agents uploading ARM data to the Management Server database, the scheduled data persistence may take more than a few minutes. As disk access may be a bottleneck for persisting or retrieving data to/from the DB, make sure the hard drive and the disk interface have good read/write performance. Consider keeping the database on a dedicated physical disk if possible and using RAID.
- In a large-scale environment, we suggest increasing the Maximum Heap size for the WebSphere Application Server 5.0 JVM where the Management Server runs.

  From the WebSphere Application Server admin console, select **Servers** → **Application Servers** → **server1** → **Process Definition** → **Java Virtual Machine**, and set the Max heap Size to 256 > Larger Value.

  Consider changing the WebSphere Application Server JVM Maximum Heap size to half the physical memory on the system if there are no competing products that require the unallocated memory.

  Note: Having a higher setting for the WebSphere Application Server JVM Maximum Heap size means that WebSphere Application Server can use up to this maximum value if required.

- Run db2 reorgchk daily on the database to prevent the UI/Reports performance from degrading as the database grows. This command will reorganize the indexes.

  Note: The db2 **`reorgchk`** command might take some time to complete and may need to be scheduled at off peak times.

Best practice for J2EE application monitoring and debugging

Out of the box, the TMTP J2EE Monitoring Component records a summary of the transactions in the J2EE application server. This default summary level is optimal

for long term monitoring during normal operation. The default settings include the following characteristics:

- Only record aggregate records
- 20% sampling rate
- Low tracing detail

With these settings, the normal transaction flow is recorded for 20% of the actual user transactions and only a summary or aggregate of the data is saved. The Low trace level turns on tracing for all inbound HTTP requests and all outbound JDBC and RMI requests. This setting allows for minimal performance impact on the monitored application server while still providing informative real time and historical data.

However, when a performance or availability violation is detected for the J2EE application server, it may become necessary to turn up some of the tracing detail to allow for quick debugging of the situation. This can easily be done by editing the existing Listening Policy and, under the section Configure J2EE settings the J2EE Trace Detail Level to Medium or High. Figure 6-9 shows how to change the default J2EE Trace Detail Level.

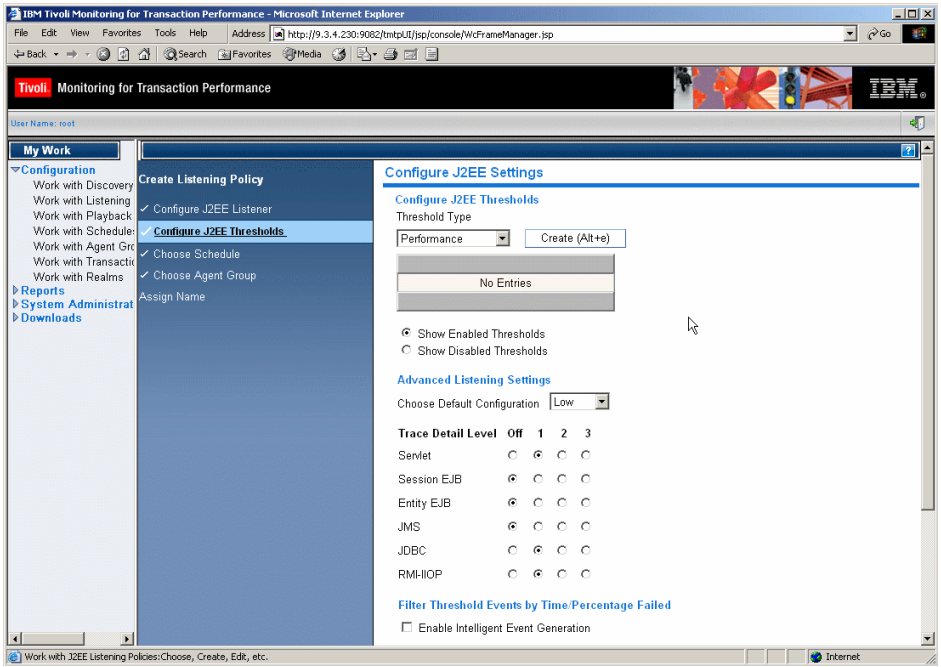


Figure 6-9 Configuring the J2EE Trace Level

The next time a violation occurs on that system, the monitoring component will automatically switch to collect instance data at its higher tracing detail. Customers with high traffic Web sites should set the sample rate lower than 20% and specify the maximum number of instances after failure on the Configure J2EE Listener page. Figure 6-10 shows how to set Sample Rate and specify the maximum number of Instances after failure.

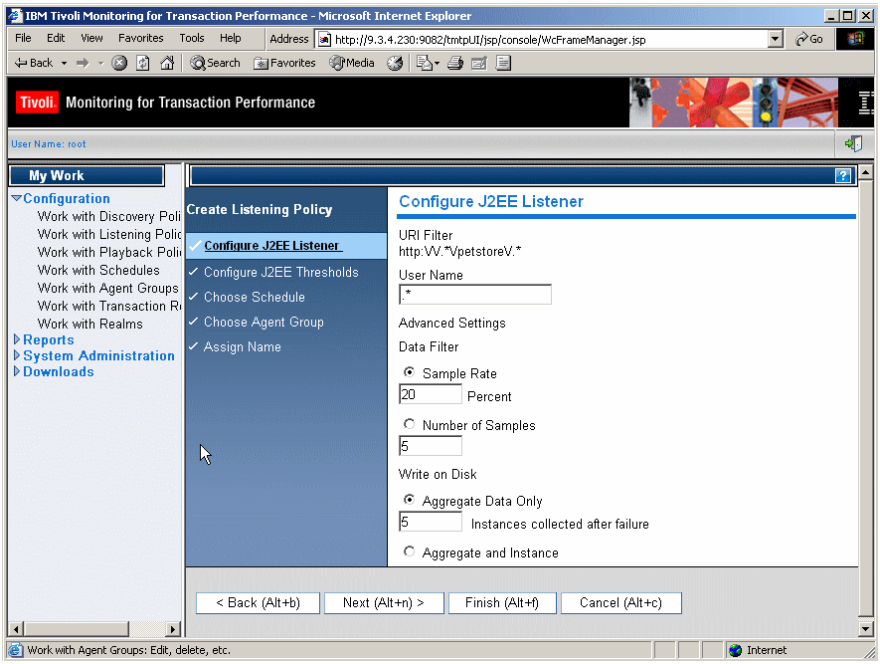


Figure 6-10 Configuring the Sample Rate and Failure Instances collected

This approach is recommended instead of manually changing the policy to collect Aggregate and Instance records. Collecting both Aggregate and full instance records has the potential to produce significant amounts of data that may not necessarily be required at normal operating levels. If you allow the Management Agent to dynamically switch to instance data collection when a violation occurs, then your instance records will only contain situations that resulted in the violation. With the higher J2EE Trace Detail Level, more transaction context information will be collected. Therefore, it will incur larger overhead on the instrumented J2EE application server. There are also larger amounts of data to be uploaded to the Management Server and persisted in the database. As a result, it may take a longer time to retrieve the latest data from Big Board.

You can now drill down into the topology for the violating policy and view the instance records that violated with the highest J2EE tracing detail. You can see exactly which J2EE class is performing outside its threshold and view its metric data to see what it was doing when it violated.

Once you have finished debugging the performance violation, it is recommended that the Listening Policy be changed to its default trace level of Low so that a minimal amount of data is collected at normal operation levels. This will improve the performance of the monitored J2EE application server and reduce the amount of data to be rolled up to Management Server.

Running DB2 on AIX

- Do not create a 64-bit DB2 instance if you intend to use TEDW 1.1, as the DB2 7.2 client cannot connect to a 64-bit database.
- Make sure to select Large File Enabled during the file system creation, so it can support files larger than 2 GB in size.
- While performing large scale testing, we found that creating a file system of 14 GB in size to accommodate the TMTP DB was sufficient.
- The database instance owner must have unlimited file size support. DB2 defaults to this, but double check in /etc/security/olimits. The instance owner should have fsize = -1.

Part 3

Using TMTP to measure transaction performance

This part discusses the use of TMTP to measure both actual, real-time end-user as well as simulated transaction response times.

© Copyright IBM Corp. 2003. All rights reserved.

The information is divided into the following main sections:

Chapter 7, “Real-time reporting” on page 211

This chapter introduces the reader to the various reporting options available to users of TMTP, both real-time and historical.

- Chapter 8, “Measuring e-business transaction response times” on page 225

  This chapter focuses on how to set up and deploy TMTP to capture real-time experiences as experienced by the end users.

  Real-time end-user measurement by Quality of Service and J2EE are introduced, and the use of subtransaction analysis and back-end service time from Quality of Service are demonstrated along with the use of correlation of the information to identify the root cause of e-business transaction problems.

- Chapter 9, “Rational Robot and GenWin” on page 325

  This chapter demonstrates how to use the Rational Robot to record e-business transactions, how to instrument those transactions in order to generate relevant e-business transaction performance data, and how to use TMTP’s GenWin facility to manage playback of your transactions.

- Chapter 10, “Historical reporting” on page 375

  This chapter discusses methods and processes of collecting business transaction data from the TMTP Version 5.2 relational database to Tivoli Enterprise Date Warehouse, and analysis and presentation of that data as a business point of view.

The target audience for this part are the users of IBM Tivoli Monitoring for Transaction Performance who are responsible for defining monitoring policies and interpreting the results.

7

Real-time reporting

This chapter introduces the various reporting options available in IBM Tivoli Monitoring for Transaction Performance Version 5.2, both real time and historical. Later chapters build on the information introduced here in order to show real e-business transaction performance troubleshooting techniques using TMTP.

© Copyright IBM Corp. 2003. All rights reserved.

7.1 Reporting overview

The focus of IBM Tivoli Monitoring for Transaction Performance reporting is to help pinpoint problems with transactions defined in monitoring policies by showing how each subtransaction relates in the overall transaction, and how those transactions compare against each other. Two main avenues are provided for viewing the data, from the Big Board, with its associated topologies and line charts, through the General Reports link, which offers additional line charts and tables. The Big Board is greatly expanded from the Big Board in 5.1 and includes access to much more data and provides greater interactivity. The primary report is the Topology View, which shows the path of a transaction throughout the system. The other reports provide additional context and comparison to the transactions behavior.

7.2 Reporting differences from Version 5.1

There are a number of reporting differences between Version 5.2 and Version 5.1 of IBM Tivoli Monitoring for Transaction Performance Web Transaction Performance. Most of the changes are good; however, a couple introduce differences that need to be understood by users familiar with previous versions.

Among the better changes are:

- Version 5.2 now makes the Big Board the focus of reporting. When problems arise, TMTP Version 5.2 users are expected to access the Big Board first, as it enables them to quickly focus on the potential problem cause.
- The other reports are for either daily reporting or to gain extra context into problems:
  - What is the behavior of this policy over time?
  - What were my slowest policies last week?
  - What is the availability of this policy in the last 24 hours?
- The Topology Report is a completely new way of visualizing the transaction. The customer can now visually see the performance of a transaction for both specific transaction instances as well as an hourly, aggregate view.
- In addition to performance and response code (availability) thresholds, the topology has “interpreted” status icons for subtransactions that might be behaving poorly. This is especially true when looking at instance topology, where the user can compare subtransaction times to the average for the hour to help determine under-performing transactions.

Other changes which users experienced with previous versions need to be aware of are:

- The STI graph (bar chart) is now based off of hourly data instead of instance data. For a policy running every 15 minutes, that means only one bar per hour. Drilling down into the STI data for the hour's topology shows a drop-down list of each instance.
- QoS graphs are hourly now instead of the former one minute aggregates.
- While not a reporting limitation, data is only rolled up to the server every hour causing the graphs to not update as quickly as before. However, a user can force an update by selecting the **Retrieve Latest Data**. The behavior of this function is explained in further detail in the following sections.
- Page Analyzer Viewer is no longer linked from the STI event view. Page Analyzer Viewer data is only accessible through the Page Analyzer Viewer report, where you choose an STI policy, Management Agent, and time.
- There is no equivalent to the QoS report with all the hits to the QoS system in one minute. However, if the collection of instance data is turned on, which is not the default, all QoS data may be viewed through the instance topologies.

7.3 The Big Board

The Big Board provides a quick summary of the state of all active monitoring policies with policy status being determined by thresholds defined by the user or generated based on the automatic baselining capabilities incorporated into the product. Please refer to 8.3, "Deployment, configuration, and ARM data collection" on page 239 for a description of the automatic baselining and thresholding capabilities of TMTP Version 5.2. Figure 7-1 on page 214 shows an example of the Big Board with transactions failing, violating thresholds, and executing normally.

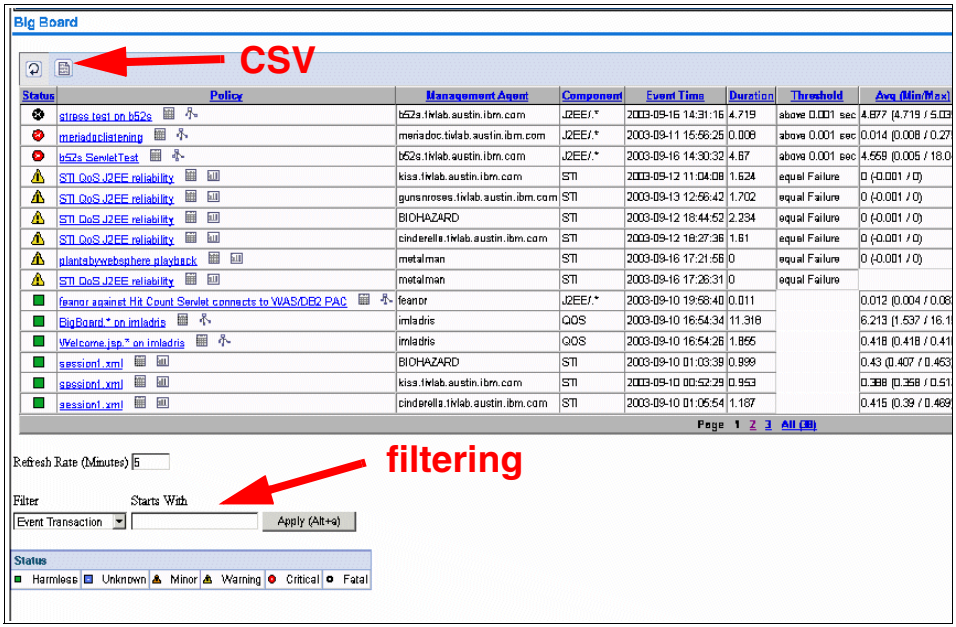


Figure 7-1 The Big Board

Event data updates the values for duration, time, and transactions as thresholds are breached. Those values are shown as columns. Uploaded aggregate data are used to update the Average (Min/Max) column so that even if there is no event activity, the row is changing. Clicking the monitoring policy name displays a summary table describing the policy's details, while clicking the **Event** icon displays a table with all the events for that policy.

Table 7-1 Big Board Icons

Icon	Description
	Display transaction events
	Display STI graph
	Display Topology View
	Export to CSV file
	Refresh view

The Big Board provides two entry points into further reporting. The first is by clicking on the **Display STI graph** icon, where you are taken to the STI Bar chart view. The second is accessed by clicking on the **Display Topology View** icon, which brings you to the Topology View.

A refresh rate may be set, and stored in the user's settings, to update the Big Board at a certain interval. Users also have the option of clicking on the **Refresh View** icon to manually refresh the view.

The Big Board's columns may be filtered by entering criteria into the drop-down box at the bottom of the dialog and choosing a column to filter. The filtering is done by finding all the columns that start with the letters entered in the text field.

Data may be exported from the Big Board by clicking on the **Export to CSV** icon.

7.4 Topology Report overview

The Topology Report provides a breakdown view of a transaction as encountered on the system. It shows hourly averages of the transactions (called aggregates) for each policy, with options to see specific instances for that hour, if enabled in the policy. Each box shown in Figure 7-2 on page 216 represents a node, and also provides a flyover with the specific transaction name and further data about the transaction.

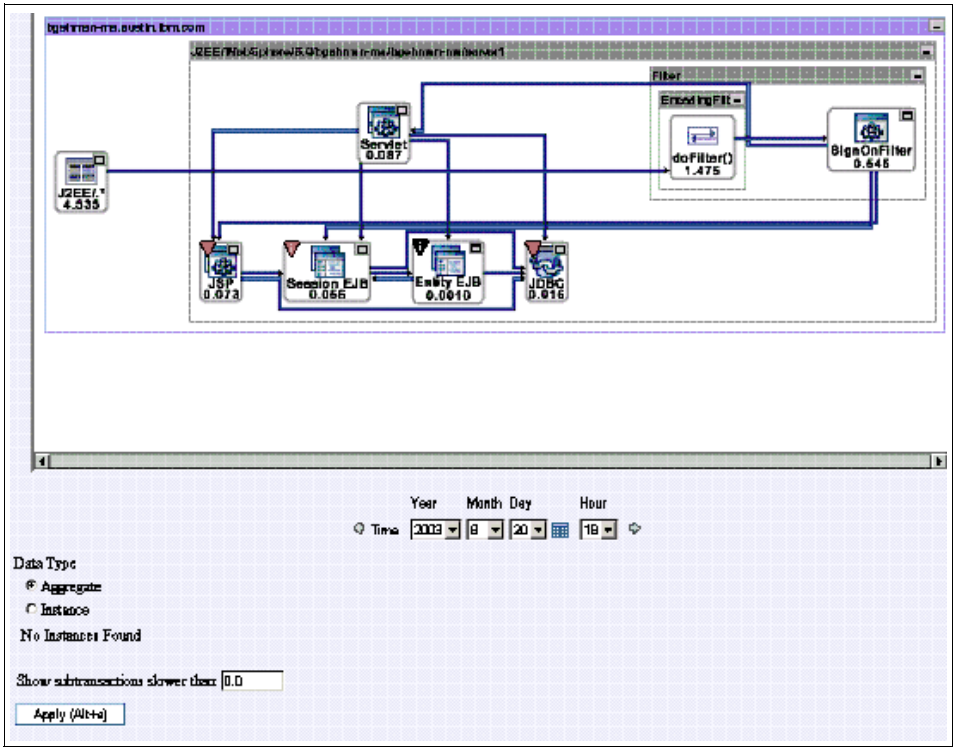


Figure 7-2 Topology Report

The Topology Report can provide topologies for any application data, though the J2EE topologies have the most subtransactions.

Data within the Topology Report is grouped into four or more types of nested boxes:

- Hosts
- Applications
- Types
- Transactions

If the nodes group has had a violation, then there will be a color coded status icon that indicates the severity of the violation.

From within the Topology Report, five additional views are available via a right-click menu, as shown in Figure 7-3 on page 217:

Event View A table of the policy events for that hour.

Response Time View
: An hourly averages over time line chart for the chosen node.

Web Health Console
: Launch the ITM Web Health console for the endpoint.

Thresholds View
: View and create a threshold for the chosen node's transaction name.

Min/Max View
: View a table of metric values (context information) for the minimum and maximum instances of that node for the hour. This report is only available from the aggregate view.

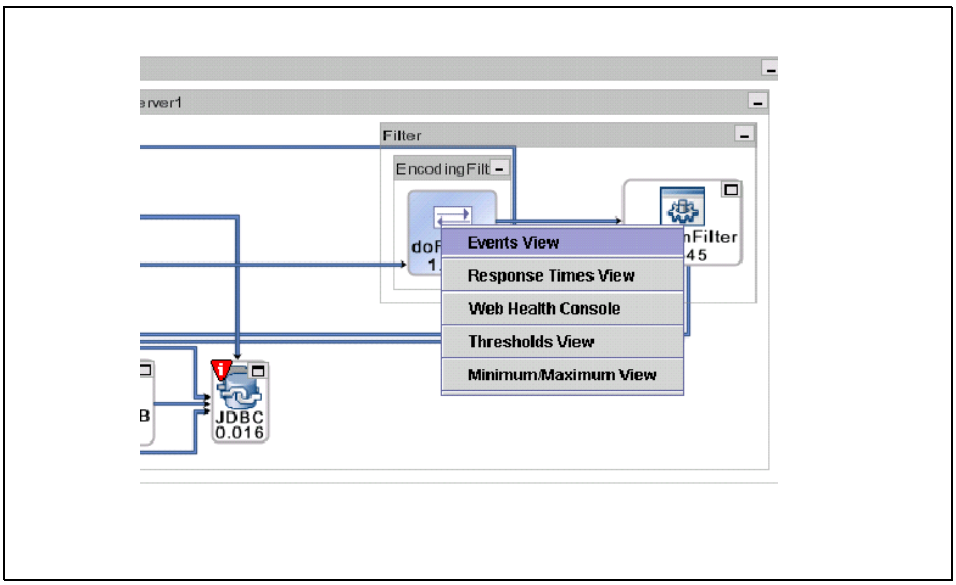


Figure 7-3 Node context reports

Examining specific instances of a transaction can be enabled during the creation of the policy, or can occur after a violation of a threshold on the root transaction.

Instance topologies are reached by choosing the instance radio button on the Aggregate View and the instance in the list and clicking the **Apply** button.

Node's status icons are set to the most severe threshold reached or compared to the average for the hour, and if the time greatly exceeds the average a more severe threshold is set. These comparisons to the average are sometimes called the *interpreted status* and are useful because they show the slow transactions helping pinpoint the cause of the problem.

Line chart from Topology View

The line chart is viewed by choosing **Response Times View** from the Topology View. By default, this shows data for the chosen node from the past 24 hour period, showing the behavior of the node over long periods of time.

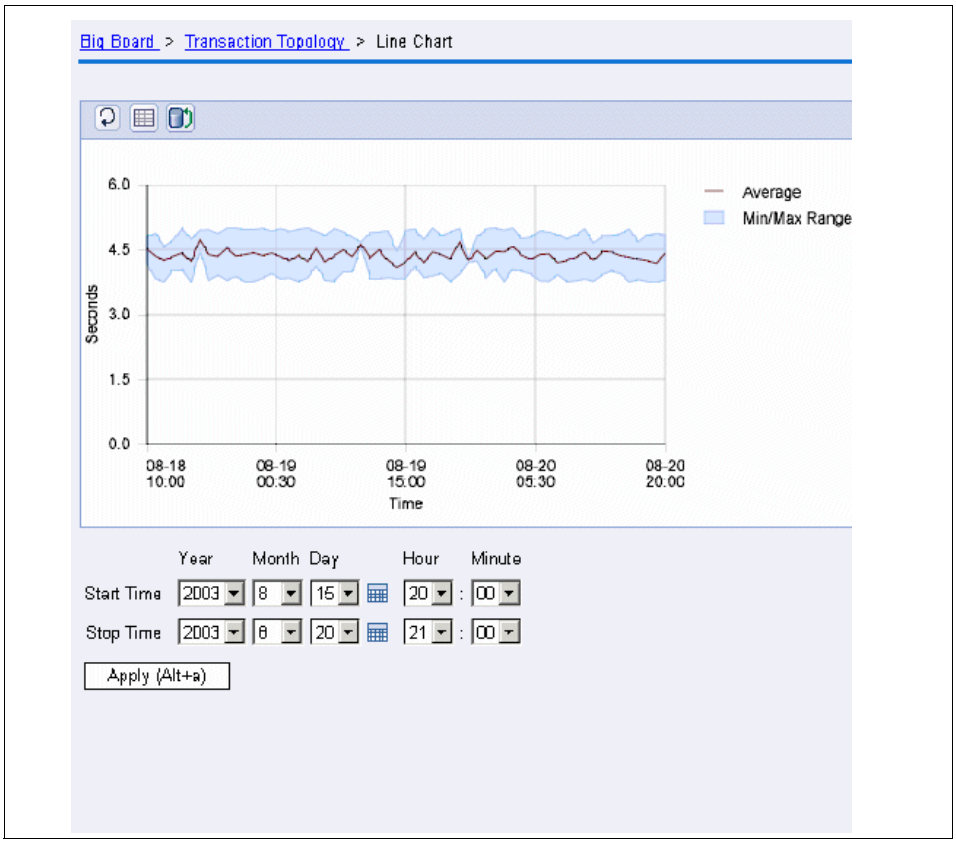


Figure 7-4 Topology Line Chart

The main line shown in the sample Topology Line Chart shown in Figure 7-4 represents the hourly averages for the node, while a blue shaded area represents the minimum and maximum values for those same hours.

If the time range is for 24 hours or less, then each point is a hyperlink that shows the aggregate topology for that hour. If there are 25 hours or more shown, there are no points to click, but the time range can be shortened around an area of interest to provide access to these topologies.

7.5 STI Report

The STI Report shows the hourly performance of the STI playback policy over time.

The initial view shows the time length of the overall transactions, which are color-coded to show if any thresholds were breached (yellow) or if there were any availability violations (red). An example of the STI Report main dialog is shown in Figure 7-5.

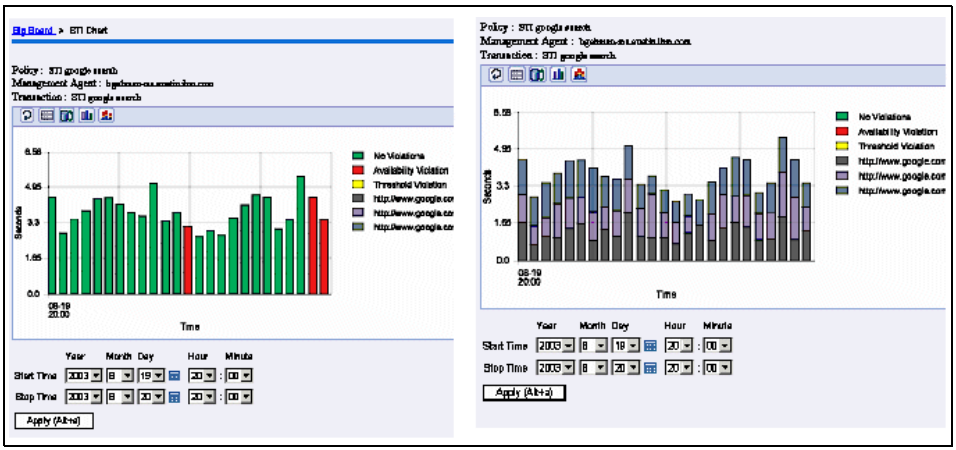


Figure 7-5 STI Reports

Clicking on any bar will decompose the bar into subsequent pieces that represent each STI subtransaction that make up the recording. This allows a comparison of the performance of each subtransaction against its peers.

Clicking any decomposed bar will take the user to the Topology View for that hour for STI.

7.6 General Reports

The General Reports option provides an entry point into reporting without going through the Big Board. This means that policies that are no longer active may have their data viewed. It provides access to six types of report:

Overall Transactions over time A line chart of endpoint(s) data plotted over time

Transactions with Subtransactions A stacked area graph of subtransactions

	compared against each other and their parent over time
Slowest transactions	A table providing the slowest root transactions in the system
General Topology	Provides topologies for all policies whether they are active or not
Availability Graph	The health of a Policy over time
Page Analyzer Viewer	Detailed breakdown of the STI transactions data

All six types of reports can be reached from the main General Reports dialog shown in Figure 7-6.

Figure 7-6 General reports

Overall Transactions Over Time

This report shows the hourly performance of an transaction for a specified policy and agents over time. It allows multiple agent's averages to be plotted against

each other for comparison. In addition, a solid horizontal line represents the policy threshold.

Transactions with Subtransactions

This report shows the hourly performance of subtransactions for a specified transaction (and policy and agent) in a stacked area graph, as shown in Figure 7-7.

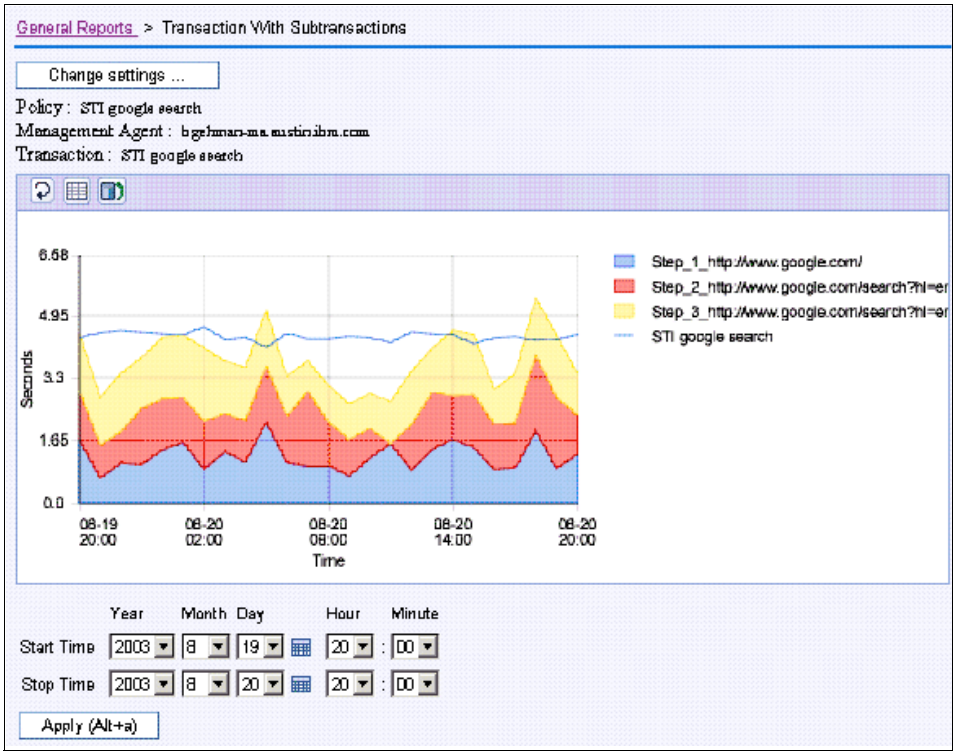


Figure 7-7 Transactions with Subtransactions report

Up to five subtransactions can be viewed for the selected transaction. By default, the five subtransactions with the highest average time will be displayed.

The legend depicting each subtransaction can be used (via clicking) to enable or disable the display of a particular subtransaction to show how its performance is affecting the transaction performance.

This is the only general report where subtransactions are plotted over time; the only other place to get this information is from the Topology Node view.

Slowest Transactions Table

This report list the worst performing transactions either for the entire Management Server or a specific application. The table shows the recent hourly aggregate data available for each root. The report allows you to choose the number of transactions to display, ranging between 5 and 100. Links are provided to the relevant topology or STI bar chart, similar to the ones in the Big Board.

General Topology

Presents the same information that is available through the Big Board's Topology View, but this report offers flexibility in changing which Listening/Playback policy to show the data for. This allows older, no longer active data to be viewed in addition to any currently active policies. All other behaviors for line charts, instance topology views, and so on, are the same.

Availability Graph

Shows the health of the chosen monitoring policy as a percentage over time.

The line represents the number of failed (that is, availability violations) transactions per hour expressed as a percentage (Figure 7-8).

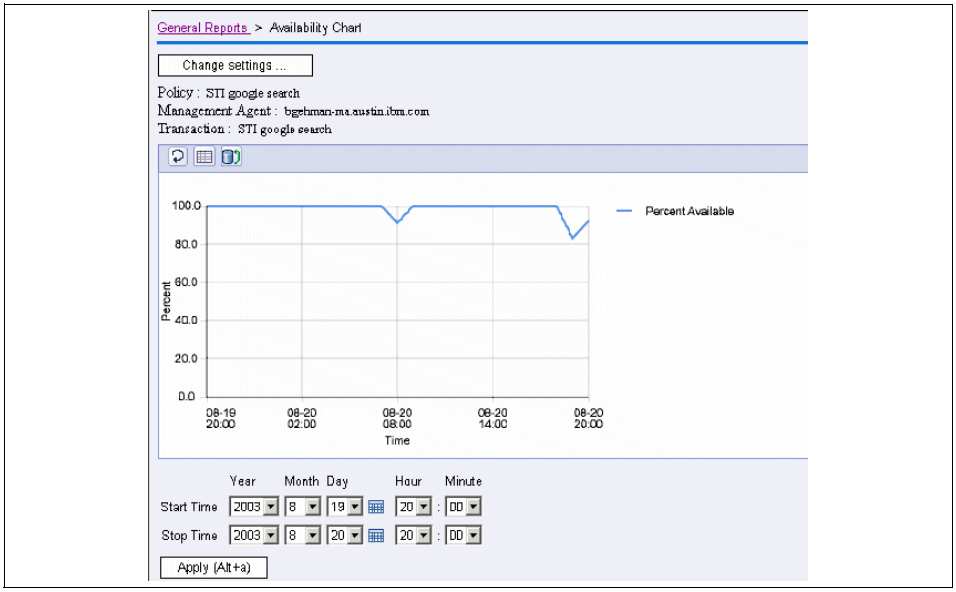


Figure 7-8 Availability graph

Page Analyzer Viewer

The Page Analyzer Viewer is the same data display mechanism as in TMTP Version 5.1 and provides a breakdown of Web page loading when loaded through the STI.

Choices are made through drop-down boxes for the policy, agent, and time of collection.

Data is collected if the Web Detailer box is checked in the STI Playback policy.

An example of a Page Analyzer Viewer report is provided in Figure 7-9.

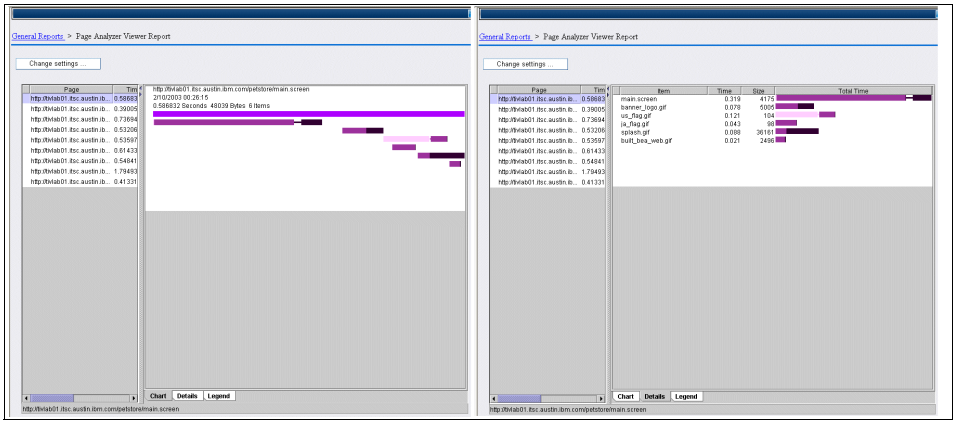


Figure 7-9 Page Analyzer Viewer

The initial view of the Page Analyzer Viewer report provides a table that lists all of the Web pages visited during the specified playback. The table columns contains the following information:

- Page displays the URL of the visited Web page.
- Time displays the total amount of time that it took to retrieve the page and render it on a Web browser.
- Size displays the number of bytes required to load the page.
- Time Stamp displays the time at which the page was visited.

With the Page Analyzer Viewer, you may also view page-specific information: to examine all of the activities and subdocuments of a visited Web page, click the name of the page in the table. A sequence of one or more bars is displayed in the right-hand pane. The bars indicate the following information:

- Bar sequence corresponds to the sequence of activities on the Web page.
- Overlapping bars indicate that activities run concurrently.

- Bar length indicates the time required for the Web page to load.
- The length of individual colored bar segments indicates the time required for individual subdocuments to load.

More detailed information about Web page activities and subdocuments can be accessed by right-clicking on a line in the chart. Using this mechanism, you can get the following information:

Idle Times	The times between Web page activities (such as subdocument loads), depicted in the chart by narrow bands between the bars in the line.
Local Socket Close	The time at which the local socket closed, depicted in the chart by a black dot.
Host Socket Close	The time at which the host socket closed, depicted in the chart by a small red caret (^) character.
Properties	A page that provides the following information about the bars in the selected line.
Summary	A summary of the number of items, connections, resolutions, servers contacted, total bytes sent and received, fastest response time (Server Response Time Low), slowest response time (Server Response Time High), and the ratio between the data points. You can use this information to evaluate connections.
Sizes	The total number of bytes that were sent and received, and the percentage of overhead for the page.
Events	A list of the violation and recovery events that were generated during page retrieval and rendering.
Comments	An area in which you can type your comments for future reference.

Lastly, by clicking on the **Details** tab at the bottom of the chart, you may see a list of the requests made by a Web page to the Web server.

8

Measuring e-business transaction response times

This chapter discusses methods and tools provided by IBM Tivoli Monitoring for Transaction Performance Version 5.2 to:

- Measure transaction and subtransaction response times in a real-time or simulated environment
- Perform detailed analysis of transaction performance data
- Identify root causes of performance problems

Real-time end-user experience measurement by using Quality of Service and J2EE will be introduced, and the use of subtransaction analysis and Back End Service Time from Quality of Service is demonstrated, along with the use of correlation of the information to identify the root cause of e-business transaction problems.

This chapter provides discussions of the following topics:

- Business and application considerations, general issues, and preparation for measurements.
- The e-business sample applications: Trade and Pet Store.

© Copyright IBM Corp. 2003. All rights reserved.

- Comparison study of choice of tools:
  - Synthetic Transaction Investigator
  - Generic Windows
  - J2EE
  - Quality of Service
- Real-time monitoring analysis using the Trade sample application in a WebSphere Application Server 5.0.1 environment using:
  - Synthetic Transaction Investigator
  - J2EE
  - Quality of Service
- Weblogic and Pet Store case study

For the discussions in this chapter, it is assumed that the TMTP Management Agent is installed on all the systems where the different monitoring components (STI, QoS, J2EE, and GenWin) are deployed. Please refer to 3.5, “TMTP implementation considerations” on page 79 for a discussion of the implementation of the TMTP Management Agent.

8.1 Preparation for measurement and configuration

Before measuring the real-time performance of any e-business application, it is very import to consider whether or not a business transaction is a candidate for being monitored, and carefully decide which data to gather. Depending on what data is of interest (User Experienced Time, Execution Time of a specific subtransaction, or total Back End Service Time are but a few examples), you will have to select monitoring tools and configure monitoring policies according to your requirements. In addition, factors related to the nature and implementation of the e-business application and your local procedures and policies may prevent you from being able to use playback monitoring tools such as Synthetic Transaction Investigator or Rational Robot (Generic Windows) because of the fact that they will generate (what to the application system seems to be) real business transactions, for example, purchases. In case you cannot back out of or cancel the transactions originating from the monitoring tool, you might want to refrain from using STI or GenWin for monitoring these transactions.

Several factors affect the decision of what to monitor, how to monitor, and from where to monitor. Some of these are:

- Use of naming standards for all TMTP policies

  To be able to clearly identify the scope and purpose of a TMTP monitoring policy, it is suggested that a standard for naming policies be developed prior to deploying TMTP in your production environment.

- Including network related issues in you monitoring data

  If you want to simulate a particular business transaction executed from specific locations in order to include network latency in your monitoring, you will have to plan for playing back the transaction from both the corporate net (intranet) and Internet in order to be able to compare end-user experienced time from two different locations. This may help you determine inexpedient routing in your network infrastructure.

  This technique may also be used to verify transaction availability from remote locations.

- Trace levels for J2EE and ARM data collection

  Depending on your level of tracing, you might incur some additional overhead (up to as much as 5%) during application execution.

  Please remember that only instances of transactions that are included in the scope of the filtering defined for a monitoring policy will incur this overhead. All other occurrences of the transaction will perform normally.

- Back-out updates performed by simulated transactions

  If Synthetic Transaction Investigator or Generic Windows is used to playback a business transaction that updates a production database with, for example, purchase orders, you might need an option to cancel or back out of the playback user's business transaction records from the database.

8.1.1 Naming standards for TMTP policies

Before creating any policies, a standard for naming discovery and listening policies should be developed. This will make it easier and more convenient for users to recognize different policies according to customer name, business application, scope of monitored transactions, and type of policy. Developing and adhering to a naming standard will especially help in distinguishing different policies and creating different type of real-time and historical reports from Tivoli Enterprise Date Warehouse.

One suggestion that may be used to name TMTP policies is:

```
<customer>_<application>_<type-of-monitoring>_<type-of-policy>
```

Using a customer name of *telia*, and application name of *trade*, the following examples would clearly convey the scope and type of different policies:

```
telia_trade_qos_lis
telia_trade_qos_dis
telia_trade_j2ee_dis
telia_trade_j2ee_lis
telia_trade_sti_forever
```

The discovery component of IBM Tivoli Monitoring for Transaction Performance enables you to identify incoming Web transactions that need monitoring. When you use the discovery process, you create a discovery policy in which you define the scope of the Web environment you want to investigate (monitor for incoming transactions). The discovery policy then samples transaction activity and produces a list of all URI requests, with average response times, that have occurred during the discovery period.

You can now consult the list of discovered URIs to identify transactions to monitor in detail using specific listening policies, which monitor incoming Web requests and collect detailed performance data in accordance with the specifications defined in the listening policy.

Defining the listening policy is the responsibility of the TMTP user or administrator responsible for a particular application area.

8.1.2 Choosing the right measurement component(s)

IBM Tivoli Monitoring for Transaction Performance Version 5.2 provides four different measuring tools, each with different capabilities and providing data that measures specific properties of the e-business transaction. The four are:

Synthetic Transaction Investigator
: Provides record and play-back capabilities for browser-based transactions. Works in conjunction with the J2EE monitoring component to provide detailed analysis for reference (pre-recorded) business transactions. STI is primarily used to verify availability and performance to ensure compliance with Service Level Objectives.

Quality of Service
: Is primarily used to monitor real-time end-user transactions, and provides user-specific data, such as User Experience Time and Round Trip Time.

J2EE
: Monitors the internals of the J2EE infrastructure server, such as WebSphere Application Server or Weblogic. Provides transaction and subtransaction data that may be used for performance, topology, and problem analysis.

Generic Windows
: Provides similar functionality as STI; however, the Rational Robot implementation allows for recording and playback of any Windows based application (not specific to the Microsoft Internet Explorer browser), but does not provide the same detailed level of data regarding times for building the end-user browser-based dialogs.

These four components may be used alone or in conjunction. Using STI or Generic Windows to play back a pre-recorded transaction that targets a URI owned by the QoS endpoint and is routed to a Web Server monitored by a J2EE endpoint will basically provide all the performance data available for that specific instance of the transaction.

The following sections provide more details that will help decide which measurement tools to use in specific circumstances.

Synthetic Transaction Investigator

TMTP STI can be used as a synthetic transaction playback and investigator tool for any Web server, such as Apache, IBM HTTP server, Sun One (formerly known as iPlanet), and Microsoft Internet Information Server, and with J2EE applications hosted by WebSphere Application Server and BEA Weblogic application servers.

Synthetic Transaction Investigator is simple to use. It is easy to record synthetic transactions and uncomplicated to run transaction playback. Compared to Generic Windows, STI playback has more robust performance measurements, simpler content checking, better HTTP response code checking, and more thorough reporting. The most important advantage is the ability of STI to instrument a HTTP request with ARM calls, thus allowing for decomposing a STI transaction in the same way that transactions monitored by the Quality of Service and J2EE monitoring components are decomposed.

Login information is encrypted.

STI is the first-choice monitoring tool, partly because it provides transaction and subtransaction response time data.

Theoretically, it is possible to use 100 STI monitoring policies inside and 100 outside the corporate network simultaneously. STI runs all the jobs in a serial fashion, which is why you should avoid running an large number of transaction performance measurements from every STI. To avoid collision between playback policies and thus ensure that all transaction response measuring tasks completes successfully, it is recommended to limit the concurrent number of tasks at a single STI monitoring component to 25 within a five minute schedule. You should also consider changing the frequency for each run of the policies from five to 10 minutes, and distribute the starting times within a 10 minute interval.

Important: The number of simultaneous playback policies you want to run depends on several factors, such as policy iteration time, the number of subtransactions in each business transaction, retry count, lap time, and timeouts.

In Version 5.2 of IBM Tivoli Monitoring for Transaction Performance the capabilities of STI have been greatly improved and now includes features like:

- Enhanced URL matching
- Multiple windows support
- Enhanced meta-refresh handling
- XML parser support
- Enhanced JavaScript support

However, despite all of these enhancements, a few limitations still apply.

Limitations of Synthetic Transaction Investigator

When working with STI, you might encounter any of the following behaviors:

Multiple windows transactions

The recorder and player cannot track multiple windows.

Multiple JavaScript requests

The recorder and player cannot process JavaScript that updates the contents of two frames. When you click the **Change frame source....** button, the newSrc()javaScript call executes function newSrc(). Example 8-1 illustrates this behavior.

Example 8-1 JavaScript call

```
{  parent.document.getElementById("myLeftFrame").src="frame_dynamic.htm"
   parent.document.getElementById("myRightFrame").src="page2.html" }
```

The content of both the left and the right frame are updated, but STI only records the first URL navigation (the one to the left frame) of the two invoked by this JavaScript.

Dynamic parameters

Certain parameters may be filled with randomly generated values at request time. For example, a HTML page containing a form element could fill at request time. A hidden input field value could be updated with a random value generated from JavaScript before the request is sent. The playback uses the result from the recorder JavaScript (it does not execute the JavaScript) when filling in the form data. This can cause incorrect data or the request to fail.

JavaScript alerts

Since the STI playback runs as a service without a user interface, the JavaScript alert cannot be answered and hangs the transaction.

Modal windows

Since the STI playback runs as a service without a user interface, the window cannot be acted upon and hangs the transaction.

Server side redirect

When a Web server redirects a page (server side redirect), a subtransaction may end prematurely and fail to process subsequent subtransactions.

Usually, the server redirect occurs on the first subtransaction. To avoid this behavior, you may initiate the recording by navigating to the server side page to which STI was redirected.

In addition, you should be aware of the following:

- Synthetic Transaction Investigator playback does not support more than one security certificate for each endpoint.
- STI might not work with other applications using a Layered Service Provider (LSP).
- STI cannot navigate to a redirected page if the Web browser running STI is configured through an authenticating HTTP proxy and a STI subtransaction is specified to a Web server redirected page. Generic Windows can be used to circumvent these problems.

Quality of Service

Quality of Service is used to provide real-time transaction performance measurements of a Web site. In addition, QoS provides metrics such as User Experienced Time, Back End Service Time, and Round Trip Time.

Note: QoS is the only measurement component of IBM Tivoli Monitoring for Transaction Performance Version 5.2 that records real-time user experience data.

Like STI, monitoring using QoS may be combined with J2EE monitoring to provide transaction breakdown and subtranaction response times for each transaction instance run through QoS. For details on how Quality of Service works, please see 3.3.1, “ARM” on page 67.

J2EE

The J2EE monitoring component is used to analyze real-time J2EE application server transaction performance and status information of:

- Servlets
- EJBs
- RMIs
- JDBC objects

J2EE monitoring collects instance level metric data at numerous locations along the transaction path. It uses JITI technology to seamless insert probes into the Java methods at class load time. These probes issue ARM calls where appropriate.

For practical monitoring, J2EE is often combines with one of the other monitoring components (typically STI or GenWin) in order to provide transaction performance measurements in a *controlled* environment. This technique is used to provide baselining and to verify compliance with Service Level Objectives for pre-recorded transactions. For real-time transactions, J2EE monitoring is primarily used for monitoring a limited number of critical subtransactions and may be activated on-the-fly to help in problem determination and identification of bottle-necks.

Details of the inner workings of the J2EE endpoint are provided in 3.3.2, "J2EE instrumentation" on page 72 and are depicted in Figure 3-8 on page 75.

Note: J2EE is the only IBM Tivoli Monitoring for Transaction Performance Version 5.2 monitoring component that is capable of monitoring the subtransaction response times within WebSphere Application Server and BEA Weblogic application servers.

Generic Windows

The Generic Windows recording and playback component in TMTP Version 5.2 is based on technology from Rational, which was acquired by IBM in 2003. Rational Robot's Generic Windows component is specially designed to measure performance and availability of Windows-based applications. Like STI, Generic Windows (GenWin) performs analysis on synthetic transactions. Like STI, GenWin can record and playback Web browser-based applications, but in addition, GenWin can record and playback any application that can run on a Windows platform, provided the application performs some kind of screen interaction.

For playing back a GenWin recorded transaction and recording the transaction times in the TMTP environment, the GenWin recording, which is saved as a VisualBasic script, has to be executed from a Management Agent, and ARM calls must be inserted manually into the script in order to provide the measurements. The advantage of this technology is that it is possible to measure and analyze the response time of specific infinitely small or large parts of an application, because the *arm_start* and *arm_stop* calls may be placed anywhere in the script. This is an excellent supplement to STI.

In addition, GenWin provides functions to monitor dynamic page strings, which is currently a limitation in the STI endpoint. For details, see "Limitations of Synthetic Transaction Investigator" on page 231.

For more details on the Generic Windows endpoint technology, please refer to 9.2, "Introducing GenWin" on page 365.

Limitations of Generic Windows

Before planning to use GenWin scripts for production purposes, you should be aware of the following limitations in the current implementation:

- GenWin runs playback in a visual mode using an automated operator type of playback. One implication of this mode of operation is that the playback systems has to be dedicated to the playback task, and that a user has to be logged on while playback is taking place. If a user, local or remote, manipulates the mouse and/or keyboard while playback is running, the playback will be interrupted.
- If delay times are not used with the recording script, the GenWin playback will fail to search the dynamic strings.
- When a transaction is recorded by GenWin, the user IDs and passwords for e-business application site login are placed in the script file as a clear text. To avoid exposing passwords in the script, it may be stored encrypted in an file (external to the script) and passed into the script at execution time. Please refer to “Obfuscating embedded passwords in Rational Scripts” on page 464 for a description on how to use this function.
- For GenWin recording and playback, you only need a single piece of Rational Robot software, in contrast to STI. Both recording and playback should not be run from the same Rational Robot, because a Playback policy might trigger playback of a prerecorded Generic Windows synthetic transaction while you are recording another transaction.

8.1.3 Measurement component selection summary

Table 8-1 summarizes the capabilities and suggested use of the four different measurement technologies available in IBM Tivoli Monitoring for Transaction Performance Version 5.2.

Table 8-1 Choosing monitoring components

Component	Operation	Advantage	Correlation with other components	Description
STI	Transaction simulation with subtransaction correlation	Simple to use	Can be combined with J2EE and QoS with correlation	Simulated end-user experience
GenWin	Transaction simulation	Can be used as a complement of STI and a Windows application	Can be combined with QoS and J2EE, but without any correlation	Simulated end-user experience

Component	Operation	Advantage	Correlation with other components	Description
QoS	Real-time Page Rendering Time and Back End Service Time with correlation	First step to measure back-end application service for end-user transactions	Can be combined with STI and J2EE with correlation	Real-time end-user experience
J2EE	Transaction breakdown	Full breakdown analysis of business application (EJB, JavaServlet, Java Servlet pages, and JDBC)	Can be combined with STI and QoS with correlation	Application transaction response time and other metric data

For more details, please see 3.3, “Key technologies utilized by WTP” on page 67.

8.2 The sample e-business application: Trade

Trade3 is the third generation of the WebSphere end-to-end benchmark and performance sample application. The new Trade3 benchmark has been re-designed and developed to cover WebSphere’s significantly expanding programming model and performance technologies. This provides a real world workload enabling performance research and verification tests of WebSphere’s implementation of J2EE 1.3 and Web Services, including key WebSphere performance components and features.

Note: You can download Trade3 sample business application from

`http://www-3.ibm.com/software/webservers/appserv/benchmark3.html`

and follow the readme.html to install Trade on a WebSphere Application Server 5.0.1 application server.

Trade3 builds off of Trade2, which is used for performance research on a wide range of software components and platforms, including WebSphere, DB2, Java, Linux, and more. The Trade3 package provides a suite of IBM developed workloads for determining the performance of J2EE application servers.

Trade3’s new design enables performance research on J2EE 1.3, including the new EJB 2.0 component architecture, Message Driven Beans, transactions (1-phase and 2-phase commit), and Web Services (SOAP, WSDL, and UDDI).

Trade3 also drives key WebSphere performance components, such as DynaCache, WebSphere Edge Server, AXIS, and EJB caching.

The architecture of the Trade3 application is depicted in Figure 8-1.

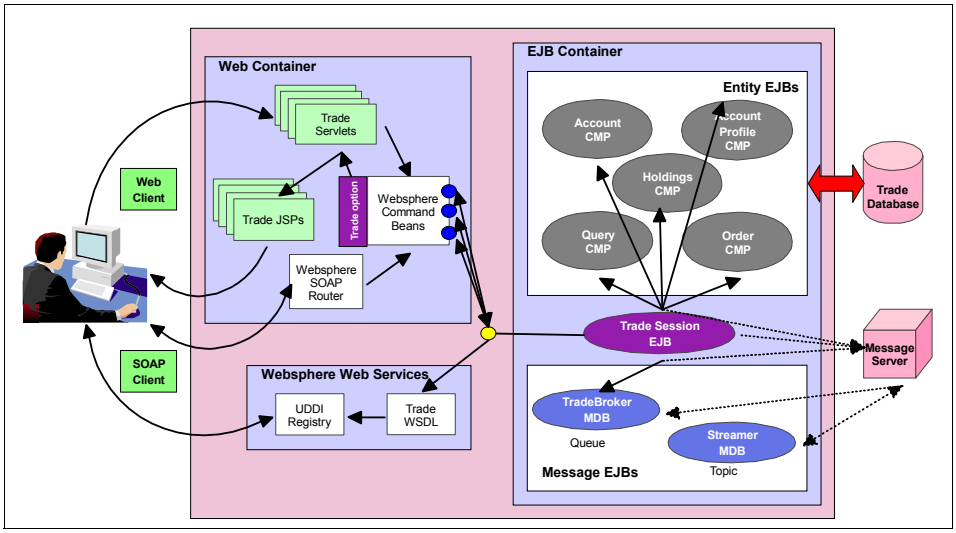


Figure 8-1 Trade3 architecture

The Trade3 application models an electronic stock brokerage providing Web and Web Services based online securities trading. Trade3 provides a real-world e-business application mix of transactional EJBs, MDBs, servlets, JSPs, JDBC, and JMS data access, adjustable to emulate various work environments. Figure 8-1 shows high-level Trade application components and a model-view-controller topology.

Trade3 implements new and significant features of the EJB 2.0 component specification. Some of these include

CRM Container Managed Relationships (CRM) provides one-to-one, one-to-many and many-to-many object to relational data managed by the EJB container and defined by an abstract persistence schema. This provides an extended, real world data model with foreign key relationships, cascaded updates/deletes, and so on.

EJB QL Standardized, portable query language for EJB finder and select methods with container managed persistence.

Local/Remote I/Fs Optimized local interfaces providing pass-by reference objects and reduced security overhead WebSphere

provides significant features to optimize the performance of EJB 2.0 workloads. These features are listed here and leveraged by the Trade3 performance workload. Performance of these features is detailed in Figure 8-1 on page 236.

EJB Data Read Ahead

A new feature of the WebSphere Application Server 5.0 persistence manager architecture for performance is various optimizations to minimize the number of database roundtrips by reading ahead and caching object structures in order to avoid round trips.

Access Intent Entity bean run-time data access characteristics can be configured to improve database access efficiency (includes access type, concurrency control, read-ahead, collection scope, and so on)

Extended EJB QL WebSphere provides critical support for extended features in EJB QL, such as aggregate functions (min, max, sum, and so on). The extended addition also provides dynamic query features.

To see the Trade application component details (as shown in Figure 8-2 on page 238), log in to:

```
https://hostname:9090/admin/
```

and click **Application** → **Enterprise Applications** → **Trade**.

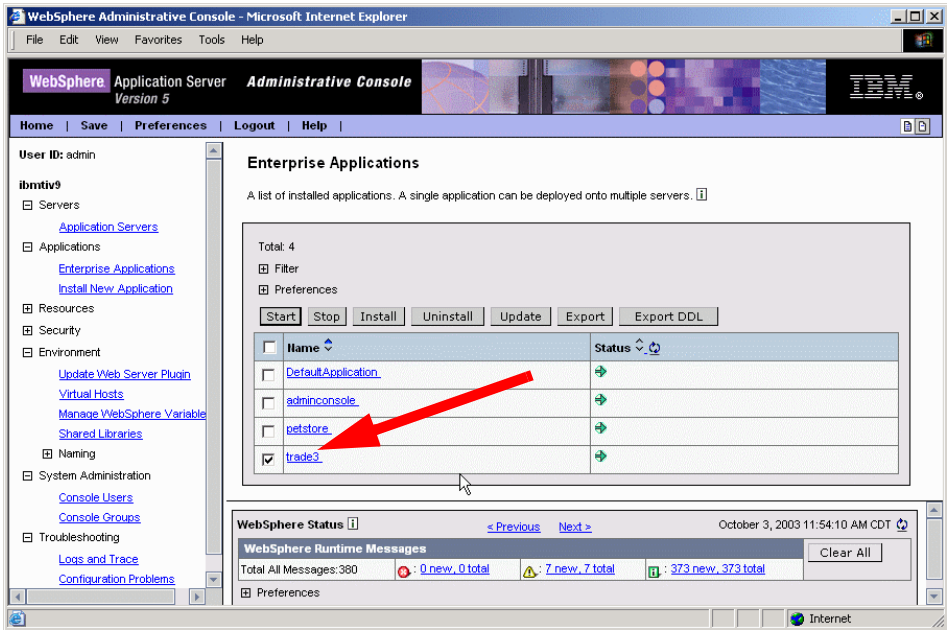


Figure 8-2 WAS 5.0 Admin console: Install of Trade3 application

In addition to a login page that is used to access the Trade system, a main home page that details the users account information and current market summary information is provided. From the user's home page, the following asynchronous transactions are processed:

- Purchase order is submitted.
- New "Open" order is created in DB.
- The new order is queued for processing.
- The "open" order is confirmed to the user.
- The message server delivers the new order message to the TradeBroker.
- The TradeBroker processes the order asynchronously, completing the purchase for the user.
- The user receives confirmation of the completed Order on a subsequent request.

8.3 Deployment, configuration, and ARM data collection

There are four different type of components that can deployed to a single Management Agent. It is possible of deploy all four components to the same system. They are:

- Synthetic Transaction Investigator
- Quality of Service
- J2EE
- Generic Windows

Once deployed, monitoring is activated by configuring and deploying different sets of monitoring specifications, known as policies, to one or more Management Agents. The monitoring policies include specifications directing the monitoring components to perform specific tasks, so the specific monitoring component referenced in a policy has to have been deployed to a Management Agent before the policy can be deployed.

IBM Tivoli Monitoring for Transaction Performance Version 5.2 operates with two types of policies:

Discovery policy

The discovery component of IBM Tivoli Monitoring for Transaction Performance enables identification of incoming Web transactions that may be monitored. When using the discovery process, a discovery policy is created, and within the discovery policy an area of the Web environment that is under investigation is specified. The discovery policy then samples transaction activity from this subset of the Web environment and produces a list of all received unique URI requests, including the average response times that were applied during the discovery period. The list of discovered URIs may be consulted in order to identify transactions that are candidates for further monitoring.

Listening policy

A listening policy collects response time data for transactions and subtransactions that are executed in the Web environment. Running a policy produces detailed information about transaction and subtransaction instance response times. A listening policy may be used to assess the experience of real users of your Web sites and to identify performance problems and bottlenecks as they occur.

Automatic thresholding

IBM Tivoli Monitoring for Transaction Performance Version 5.2 implements a new concept of automatic thresholding in both discovery and listening policies. Every node on a topology (group nodes as well as the final-click nodes) has a timing value associated with it. The final-click node's timings will stay the same, but the group node's timings will now be the maximum timing contained within that group.

The worst performing overall transaction is marked *Most Violated*. A configurable percentage (default 5%) of topology nodes is marked with the *Violated* interpreted status to show other potential areas of concern. If only one node in the whole topology is to be marked, it is the Most Violated node and there will be no Violated nodes.

The Topology algorithm does not rely on timing percentages to determine what is Violated and Most Violated. Instead, it compares the absolute difference between the instance and aggregate timing data while subtracting the sum of the values of the children instances. This provides for a more accurate estimate of the worst performing subtransaction, because it is an estimate of the time actually spent in the node.

The value calculated for each node is determined by the formula:

[(sum of transaction's relations instance time) – (sum of children instance time)] – [(sum of transaction's relations aggregate time) – (sum of children aggregate average)]

This will provide a value in seconds that is an approximation of time spent in the node (method).

The transaction with the greatest of these values will be the Most Violated. The top 5% (by default) of these transactions will have status Violated. The calculated values will *not* be shown to the user. If a node has a zero or negative value when (sum of transaction's relations instance time) - (sum of transaction's relations aggregate time) occurs, then it will not be marked. The reason for this is because a negative value implies the node performed below its average for the hour, and hence cannot be considered slow.

Intelligent event generation

Enabling this option can reduce event generation. Intelligent event generation merges multiple threshold violations into a single event, making notification and reports more useful. For example, a transaction might exceed and fall below a threshold hundreds of times during a single monitoring period. Without intelligent event generation, each of these occurrences generates a separate event with associated notification.

8.4 STI recording and playback

STI measures how users might experience a Web site in the course of performing a specific transaction, such as searching for information, enrolling in a class, or viewing an account. To record a transaction, you use STI Recorder, which records the sequence of steps you take to accomplish the task. For example, viewing account information might involve logging on, viewing the main menu, viewing an account summary, and logging off. When a recorded transaction accesses one or more password-protected Web pages, you create a specification for the realm to which the pages belong. After you record a transaction, you can create an STI playback policy, which instructs the STI component to play back the recorded transaction and collect a range of performance metrics.

To set up, configure, deploy, and prepare for playing back the first STI recording, the following steps have to be completed:

1. STI component deployment
2. STI Recorder installation
3. Transaction recording and registration
4. Playback schedule definition
5. Playback policy creation

Please note that the first two steps only have to be executed once for every system that will be used to record synthetic transactions. However, steps 3 through 5 has to be repeated for every new recording.

8.4.1 STI component deployment

To deploy the STI component to an existing Management Agent, log in to the TMTP console and select **System Administration** → **Work with Agents** → **Deploy Synthetic Transaction Investigator Components** → **Go**, as shown in Figure 8-3 on page 242.

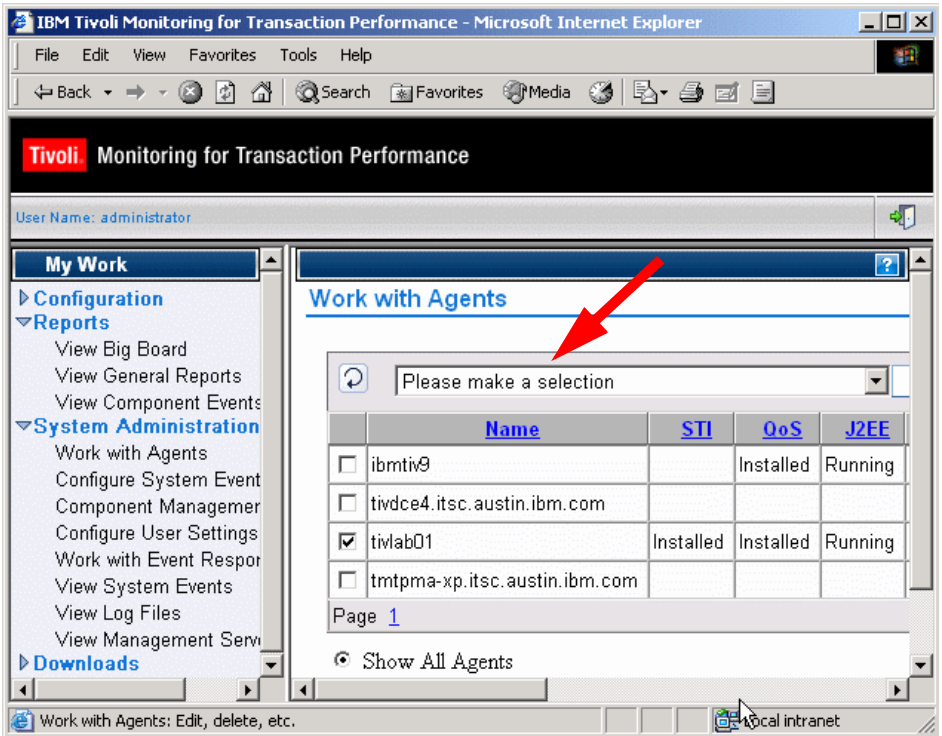


Figure 8-3 Deployment of STI components

After couple of minutes, the Management Agent will be rebooted and the Management Agent will show that STI is installed.

8.4.2 STI Recorder installation

Follow the procedure below the install the STI Recorder on a Windows based system:

1. Log in to a TMTP Version 5.2 UI console through your browser by specifying the following URL:

   `http://hostname:9082/tmtpUI/`

2. Select **Downloads** → **Download STI Recorder**.
3. Click on the **setup_sti_recorder.exe** download link.
4. From the file download dialog, select **Save**, and specify a location on your hard drive in which to store the file named setup_sti_recorder.exe.

5. When the download is complete, locate the setup_sti_recorder.exe file on your hard drive and double-click on the file to begin installation. The welcome dialog shown in Figure 8-4 will appear.

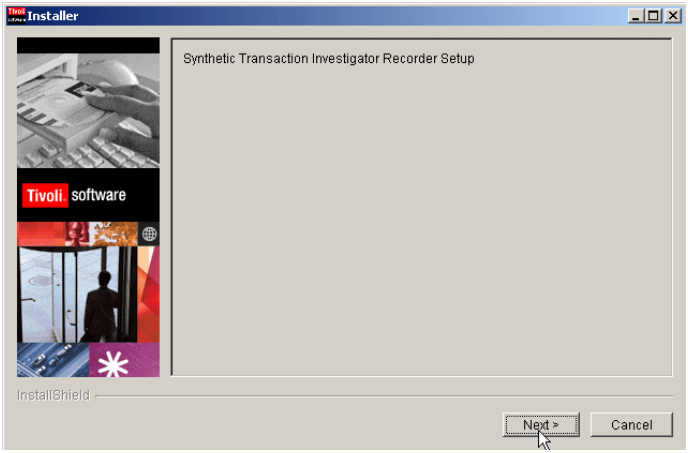


Figure 8-4 STI Recorder setup welcome dialog

6. Click **Next** to start the installation. This will make the Software License Agreement dialog, shown in Figure 8-5, appear.

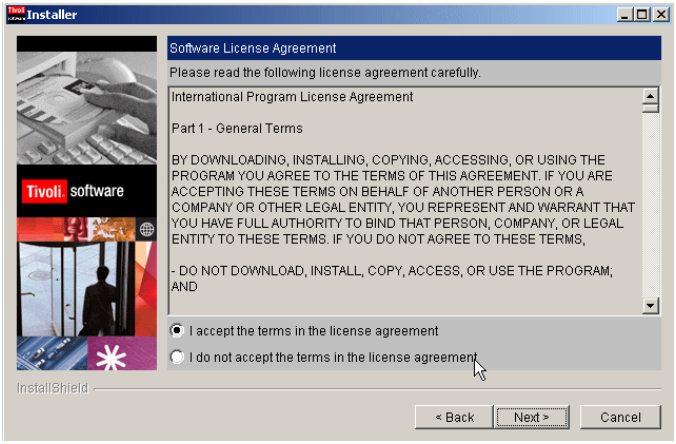

Installer
Software License Agreement
Please read the following license agreement carefully.
International Program License Agreement
Part 1 - General Terms
BY DOWNLOADING, INSTALLING, COPYING, ACCESSING, OR USING THE PROGRAM YOU AGREE TO THE TERMS OF THIS AGREEMENT. IF YOU ARE ACCEPTING THESE TERMS ON BEHALF OF ANOTHER PERSON OR A COMPANY OR OTHER LEGAL ENTITY, YOU REPRESENT AND WARRANT THAT YOU HAVE FULL AUTHORITY TO BIND THAT PERSON, COMPANY, OR LEGAL ENTITY TO THESE TERMS. IF YOU DO NOT AGREE TO THESE TERMS,
- DO NOT DOWNLOAD, INSTALL, COPY, ACCESS, OR USE THE PROGRAM; AND
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Tivoli software
InstallShield
< Back
Next >
Cancel

Figure 8-5 STI Software License Agreement dialog

7. Select the "**I accept...**" radio button, and click **Next**. Then, the installer depicted in Figure 8-6 on page 244 will be displayed.

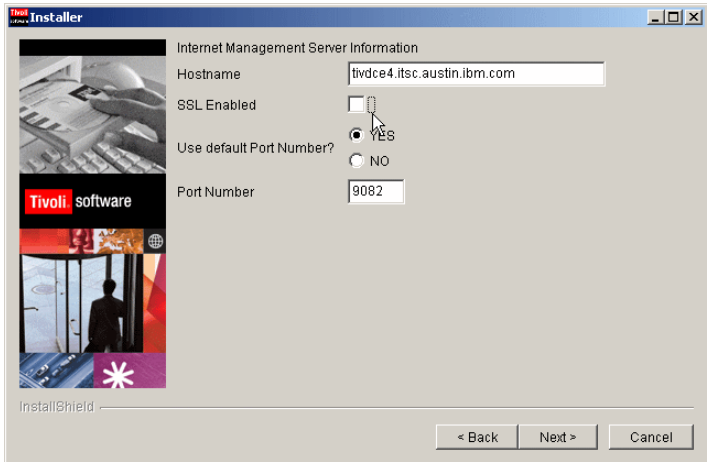


Figure 8-6 Installation of STI Recorder with SSL disable

8. Either select to enable or disable the use of Secure Socket Layer (SSL) communication. Figure 8-6 shows a configuration with SSL disabled, and Figure 8-7 shows the selection to enable SSL.

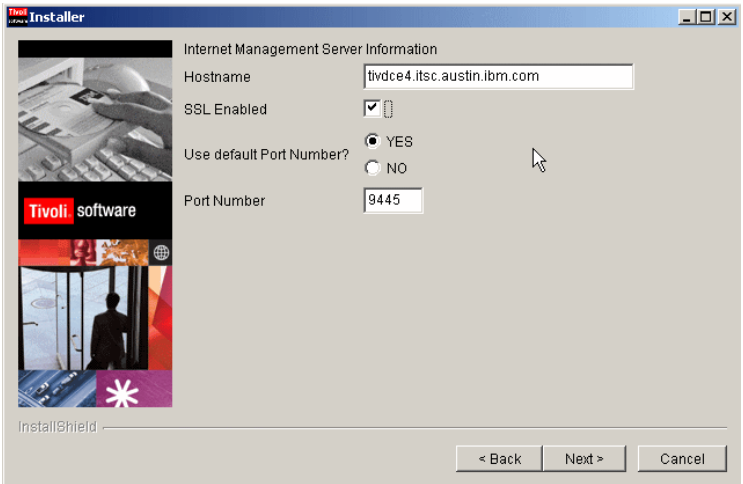


Figure 8-7 installation of STI Recorder with SSL enabled

9. Whether or not SSL has been enabled, select the port to be used to communicate with the Management Server. If in doubt, contact your local TMTP system administrator. Click **Next** and **Next**, and then **Finish** to complete the installation of the STI Recorder.
10. Once installed, the STI Recorder can be started from the Start Menu: **Start** → **Programs** → **Tivoli** → **Synthetic Transaction Investigator Recorder**,

and the setup_sti_recorder.exe file downloaded in step 4 on page 242 may be deleted.

Tip: If you want to connect your STI Recorder to a different TMTP Version 5.2 Management Server, edit the endpoint file in the c:\install-dir\STI-Recorder\lib\properties\ directory and change value of the dbmgmtsrvurl property to the host name of the new Management Server.

8.4.3 Transaction recording and registration

There are several steps involved in recording and playing back a STI transaction:

1. Record the desired e-business transaction using the STI Recorder and save it to a Management Server.
2. From your Windows Desktop, select **Start** → **Programs** → **Tivoli** → **Synthetic Transaction Investigator Recorder** to start the STI Recorder locally.
3. Type the application address in **Location** and set the Completion Time to a value that will be adequate for the transaction(s) you will be recording. Please see Figure 8-8 on page 246 for an example. When ready to start recording, press Enter.

Note: If the Completion Time is set too low, a browser action in the recording can cause STI to perform unnecessary actions or fail during playback. Setting a Completion Time that is too low is a common user error.

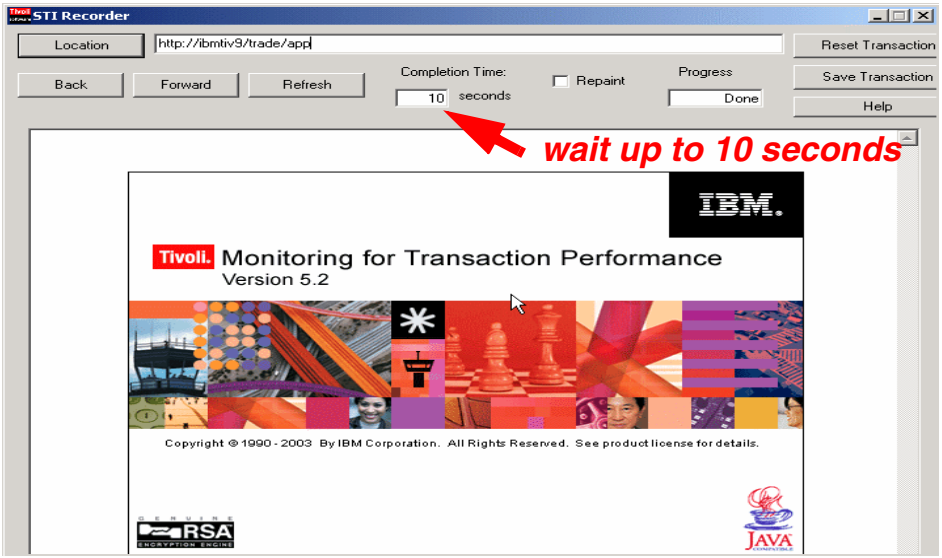


Figure 8-8 STI Recorder is recording the Trade application

4. Wait until the progress bar shows Done and start recording the desired transactions.

> **Important:** If the Web site you are recording a transaction against uses basic authentication (that is, you are presented with a pop-up window where you need to enter your user ID and password), you will need to write down the realm name, user ID and password needed for authentication to the site. This information is required in order to create a realm within TMTP. The procedure to create a realm is provided in 8.4.6, “Working with realms” on page 255.

5. When finished, press the **Save Transaction** button. Now, a XML document containing the recording is generated, as shown in Figure 8-9 on page 247.

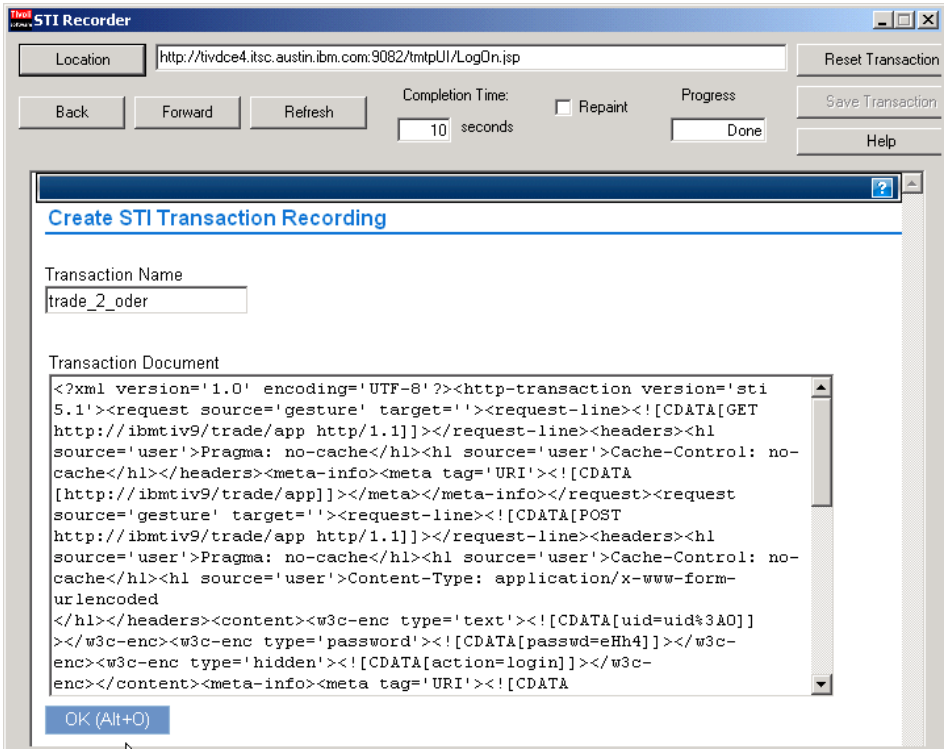


Figure 8-9 Creating STI transaction for trade

The XML document will be uploaded to the Management Server, so it can be distributed to any Management Agent with the STI component installed. To authenticate with the Management Server, provide your credentials to Management Server in order to be allowed to save the transaction with a unique name.

Once the transaction has been played back, a convenient way of getting an overview of the number of subtransactions is to look at the Transactions with Subtransactions for the STI playback policy. During setup of the report, the subtransaction selection dialog shown in Figure 8-10 on page 248 is displayed, and this clearly shows that six subtransactions are involved in the trade_2_stock-stock transaction.

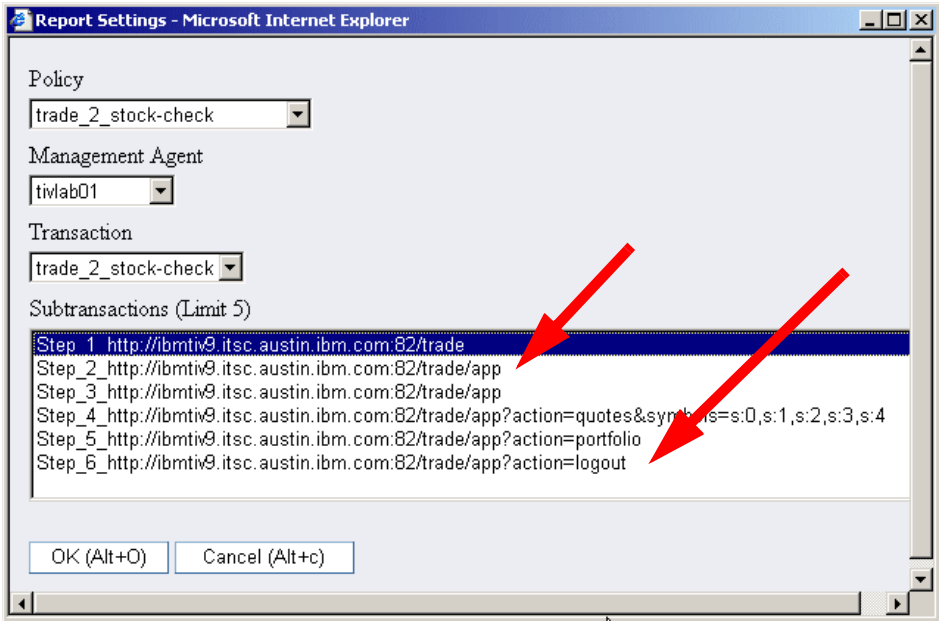


Figure 8-10 Application steps run by trade_2_stock-check playback policy

6. Click **OK** to import the XML document at the TMTP Version 5.2 Management Server.

8.4.4 Playback schedule definition

Having uploaded the STI recording, you are ready to define the run-time parameters that will control the playback of the synthetic transaction. This includes defining a schedule for the playback as well as a Listening Policy. Follow the procedure below to create a schedule for running playback policy.

1. Select **Configuration** → **Work with Schedules** → **Create New**. The dialog shown in Figure 8-11 on page 249 will be displayed.

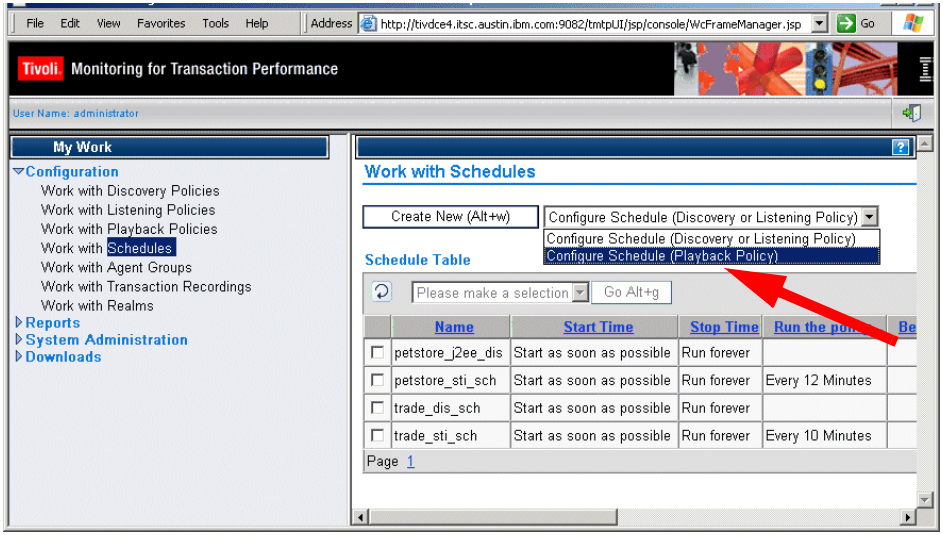


Figure 8-11 Creating a new playback schedule

Select **Configure Schedule (Playback Policy)** from the schedule type drop-down menu and press **Create New**. This will bring you to the Configure Schedule (Playback Schedule) dialog (shown in Figure 8-12 on page 250) where you specify the properties for the new schedule.

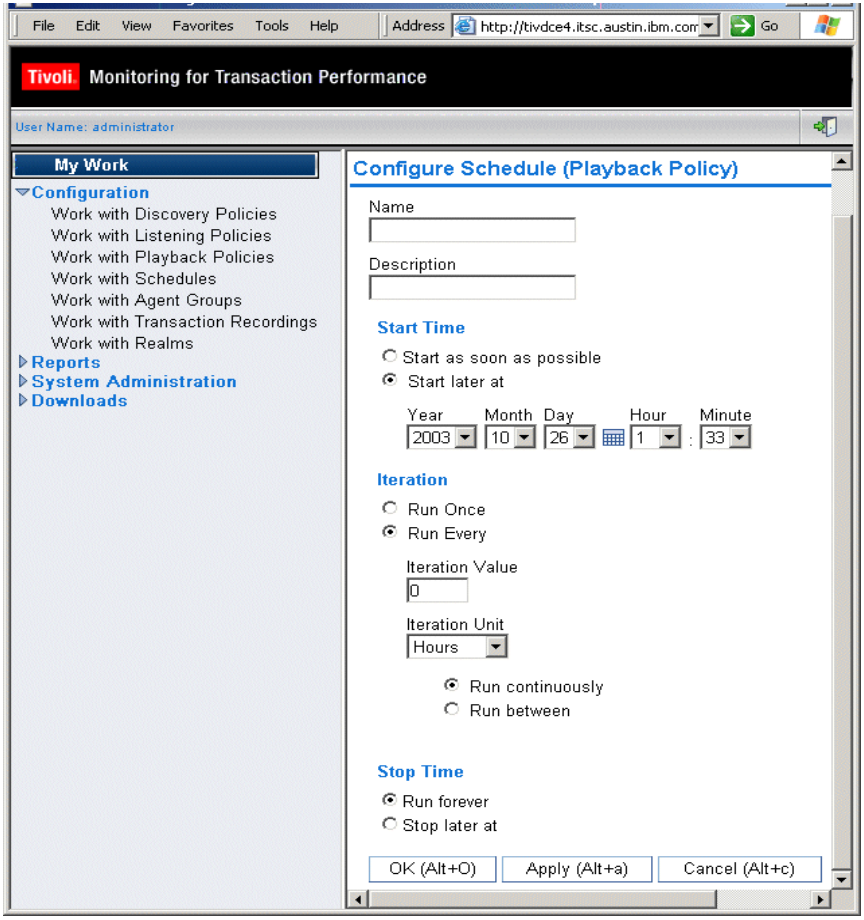


Figure 8-12 Specify new playback schedule properties

2. Provide appropriate values for all the properties of the new schedule:
   - Select a name, according to the standards you have defined, which easily conveys the purpose and frequency of the new playback schedule. For example: telia_trade_sti_15mins.
   - Set Start Time to **Start as soon as possible** or **Start later at**, depending on your preference. If you select **Start later at**, the dialog opens a set of input fields for you to fill in the desired start date.
   - Set Iteration to **Run Once** or **Run Every**. In case you choose the latter, you will be prompted for a Iteration Value and Unit.

– In case Run Every was chosen in the previous step, set the Stop Time to **Run forever** or **Stop later at**, and specify a Stop Time in case of the latter.

Press **OK** to save the new schedule.

8.4.5 Playback policy creation

After having defined a schedule (or determined to reuse one that had already been defined), the next step is to create a Playback policy for the STI recording. Follow the steps below to complete this task.

For a thorough walk-through and descriptions of all the parameters and properties specified during the STI playback definition process, please refer to the *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386.

1. From the home page of the TMTP Version 5.2 console, select **Configuration** → **Work with Playback Polices**.

   From the Work with Playback Policies dialog that is displayed (shown in Figure 8-13), set the playback type to *STI* and press the **Create New** button. Next, the Configure STI Playback dialog will appear. An example is provided in Figure 8-14 on page 252.

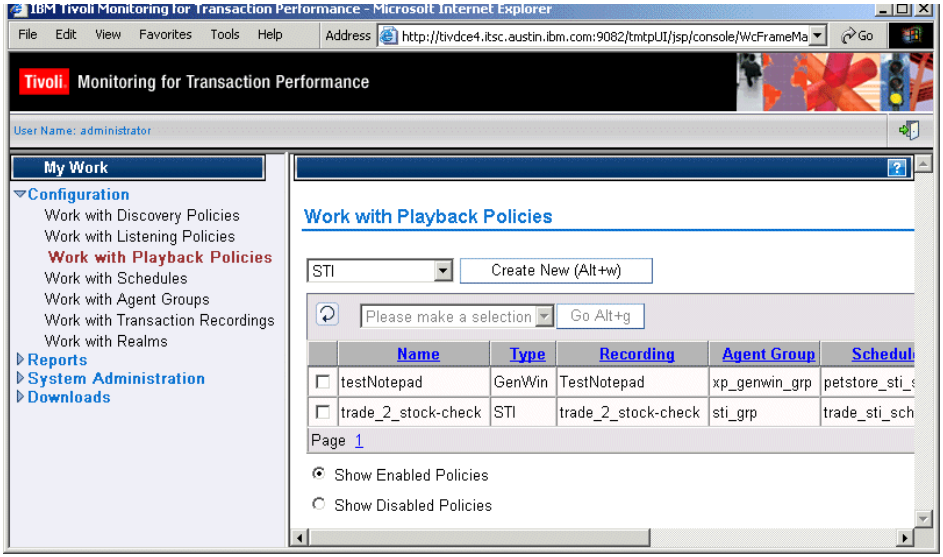


Figure 8-13 Create new Playback Policy

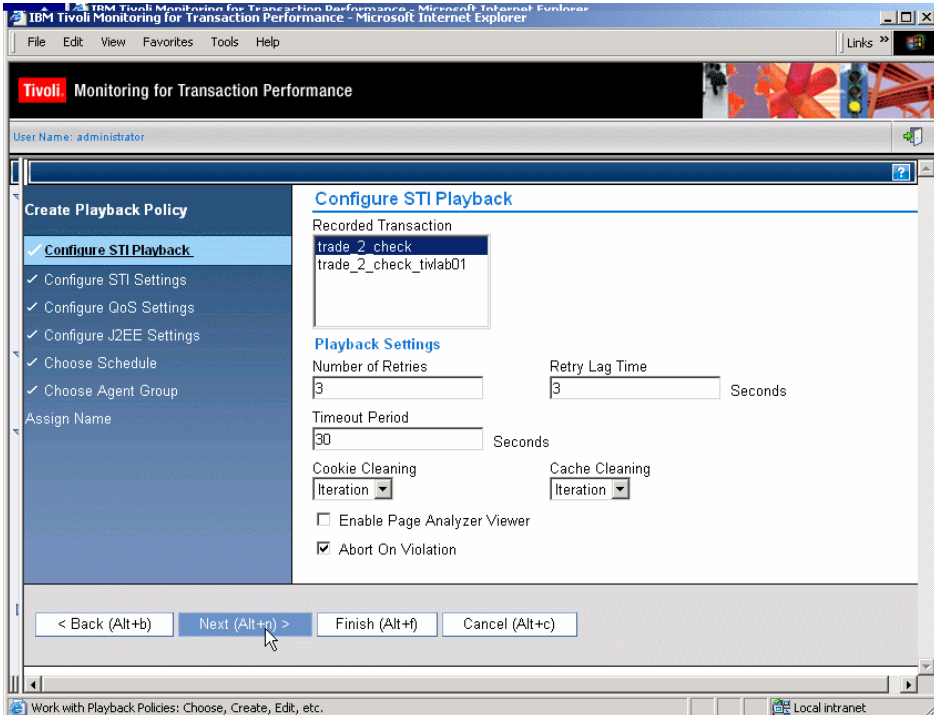


Figure 8-14 Configure STI Playback

2. Fill in the specific properties for the STI playback policy you are defining in the Create STI Playback dialogs. These are made up of seven sub-dialogs, each covering different aspects of the STI Playback. The seven subsections are:

   – Configure STI Playback
   – Configure STI Settings
   – Configure QoS Settings
   – Configure J2EE Settings
   – Choose Schedule
   – Choose Agent Group
   – Assign Name

   The following sections highlights important issues that should be aware of when defining STI playback policies. For a detailed description of all the properties, please refer to *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386.

   Please note that in order to proceed to the next dialog in the STI Playback creation chain, just click on the **Next** button at the bottom of each dialog.

– Configure STI Playback

Select the appropriate Playback Transaction, which most likely is the one you recorded and registered in the previous step described in 8.4.3, “Transaction recording and registration” on page 245.

Define the Playback Settings that applies to your transaction.

Your choices on this dialog will affect the operation and data gathering performed during playback. Some key factors to be aware of are:

- You may choose to click the **Enable Page Analyzer Viewer** for a playback. When enabled, data related to the time used to retrieve and render subdocuments of a Web page are gathered during the playback.
- By enabling Abort On Violation, you decide whether or not you want STI to abort a playback iteration if a subtransaction fails. Normally, STI aborts a playback if one of the subtransactions fails. For example, a playback is aborted when a requested Web page cannot be opened. If Abort On Violation is not enabled, STI continues with the playback and attempts to complete the transaction after a violation occurs.

Note: If a threshold violation occurs, a Page Analyzer Viewer record is automatically uploaded, even if the Enable Page Analyzer Viewer option is *not* selected. This ensures that you receive sufficient information about problems that occur.

– Configure STI settings

You can specify four different types of thresholds:

- Performance
- HTTP Response Code
- Desired Content not found
- Undesired contents found

It is possible to create multi-level performance thresholds for STI transactions and have events generated at a subtransaction level.

– Configure QoS settings

You can not create a QoS setting during the creation of a STI playback policy. However, when playback policies has been executed once (and a topology has been created), this option becomes available.

– Configure J2EE settings

If the monitored transaction is hosted by a J2EE application server, you should configure J2EE Settings using the default values as a starting point.

– Choose schedule

  Select the schedule that is defined when the STI Playback policy is executed. You may consider using the schedule created in the beginning of this section, as described in 8.4.4, “Playback schedule definition” on page 248.

– Choose agent group

  Select the group of Management Agents to execute this STI Playback policy. Please remember that the STI component has to have been deployed to each of the Management Agents in the group to ensure successful deployment and execution.

  Note: If you want to correlate STI with QoS and J2EE, choose the Agent Group where QoS and J2EE components are deployed.

– Assign Name

  Assign a name to the new STI Playback policy. In the example shown in Figure 8-15 on page 255, the name assigned is *trade_2_stock-check*.

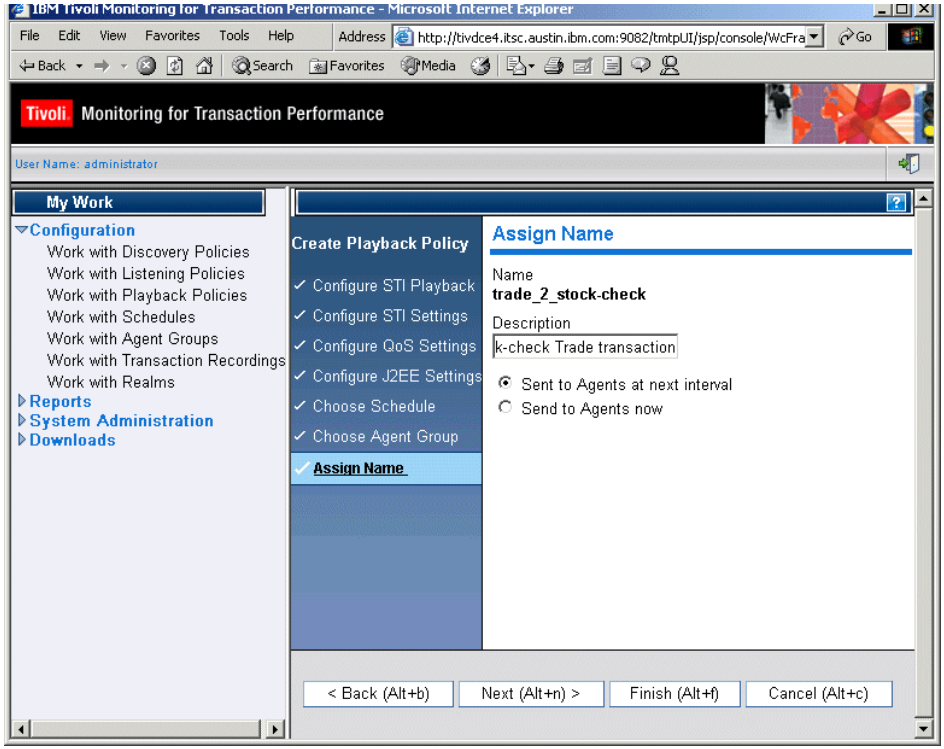


Figure 8-15 Assign name to STI Playback Policy

In addition, you can decide whether or not to distribute the STI Playback Policy to the Management Agents that are member(s) of the selected group(s) immediately, or you prefer to postpone the distribution to the next scheduled regular distribution.

Click **Finish** to complete the creation of the new STI Playback Policy.

8.4.6 Working with realms

Realms are used to specify settings for a password-protected area of your Web site that is accessed by an STI Playback Policy. If a recorded transaction passes through a password-protected realm, realm settings ensure that STI is able to access the protected pages during playback of the transaction.

Creating realms

To create a realm, click **Configuration** → **Work with Realms** → **Create New** on the home page of the TMTP Version 5.2 Management Server console. The Specify Realm Settings dialog, as shown in Figure 8-16, will appear.

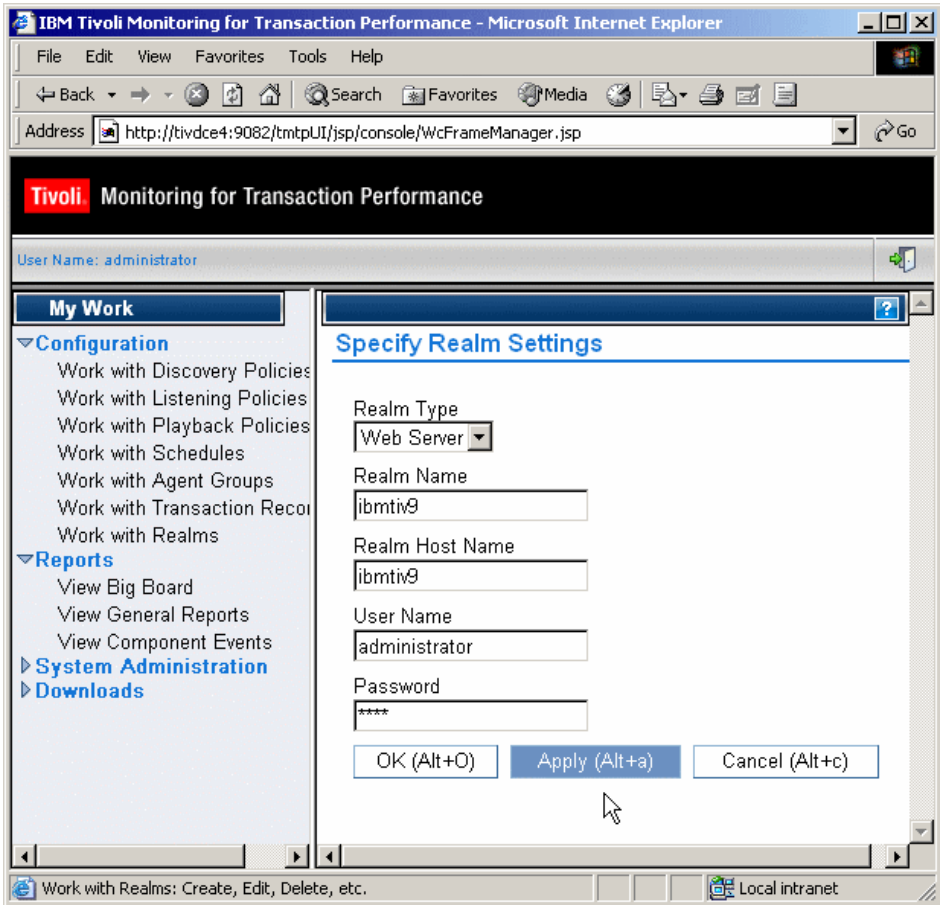


Figure 8-16 Specifying realm settings

If the transaction accesses a proxy server in a realm where a proxy server is located, choose **Proxy**. If the transaction accesses a realm where a Web server is located, choose **Web Server**.

Specify the name of the realm for which you define credentials, the fully qualified name of the system that hosts the Web site for which the realm is defined, and the User Name and Password to be used to access the realm. When finished, click **Apply**.

8.5 Quality of Service

The Quality of Service component in IBM Tivoli Monitoring for Transaction Performance Version 5.2 samples data from real-time, live HTTP transactions against a Web server and measures, among other items, the time required for the round trip of each transaction. The Quality of Service component measurements include:

- User Experience Time
- Back End Service Time
- Page Render Time

To gather this type of information, QoS intercepts the communication between end users and Web servers by means of reverse-proxy technology. This allows QoS to measure response times and to manage ARM correlators. The use of ARM allows QoS to scale better and to be incorporated with other measurement technologies, such as J2EE and STI.

When a HTTP request reaches QoS, QoS checks the request to see if the HTTP headers contain an ARM correlator from a parent transaction. If a correlator is discovered, it will consider itself to be a non-edge application (a subtransaction) in relation to gathering and recording ARM data. In case of the absence of a correlator, QoS will consider itself to be the edge application for this transaction, and generate a correlator, which is included in the HTTP request as it is passed on the server that hosts the called application.

The reverse proxy implementation provides a single entry-point to several Web servers much like a normal proxy works as an Internet gateway for multiple workstations on a corporate network, as depicted in Figure 8-17 on page 258. Without the reverse proxy, the IP addresses of all the Web servers has to be known by the requestors. With the reverse proxy, the requestors only need to know the IP address of the reverse proxy.

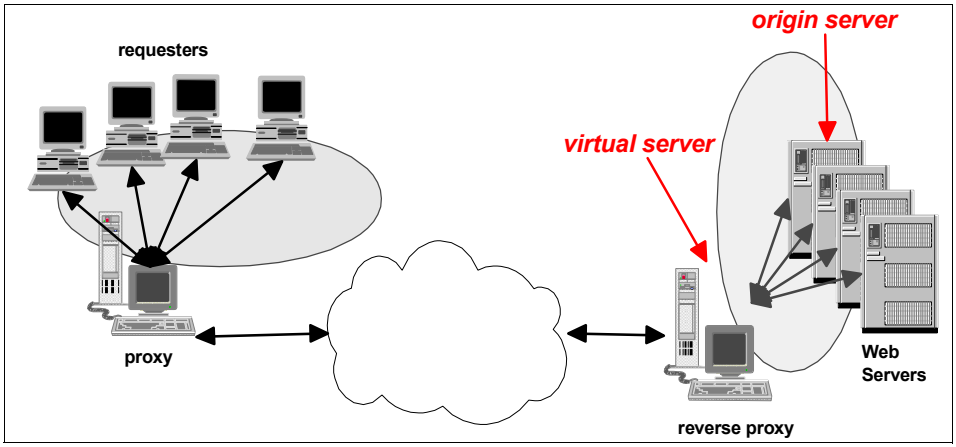


Figure 8-17 Proxies in an Internet environment

This technology is primarily implemented to circumvent some of the shortcomings of the TCP/IP addressing schema by removing the need for all servers and workstations to be addressable (known) to all other systems on the Internet, which also may be regarded as an additional security feature.

When working with the Quality of Service monitoring component, you should be familiar with the following terms:

Origin server The Web server that you want to monitor.

Proxy server A virtual server (implemented at the origin server or on a remote computer) that acts as a gateway to specific Web Servers. Normally, transactions within a Web server measures the time required to complete the transaction. This virtual server runs within IBM HTTP Server Version 1.3.26.1, which comes with the QoS monitoring component.

Reverse proxy A physical HTTP Server that hosts the virtual proxy servers pointing to the origin servers. The reverse proxy server also hosts the QoS monitoring component. The reverse proxy server may be installed directly on the origin server or on a remote computer. Running QoS on the same machine as the origin server may be beneficial, because it eliminates network issues (speed, delay, collisions, and bandwidth).

Digital certificates Authentication documents that secure communications for Quality of Service monitoring.

8.5.1 QoS Component deployment

To deploy the Quality of Service component to a Management Agent, follow the steps below:

1. From the home page of the Management Server console, click on **System Administration** → **Work with Agents**. The Work with Agents dialog depicted in Figure 8-18 will be displayed.

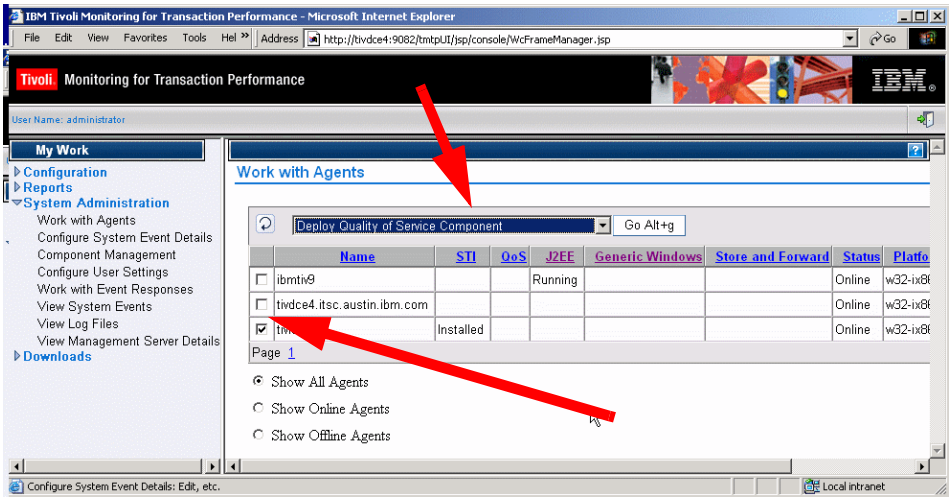


Figure 8-18 Work with agents QoS

2. Select the target to which QoS is to be deployed, and select the **Deploy Quality of Service** component from the action selection drop-down menu at the top of the Work with Agents dialog. Click **Go** to go to the configuration of the new Quality of Service component.

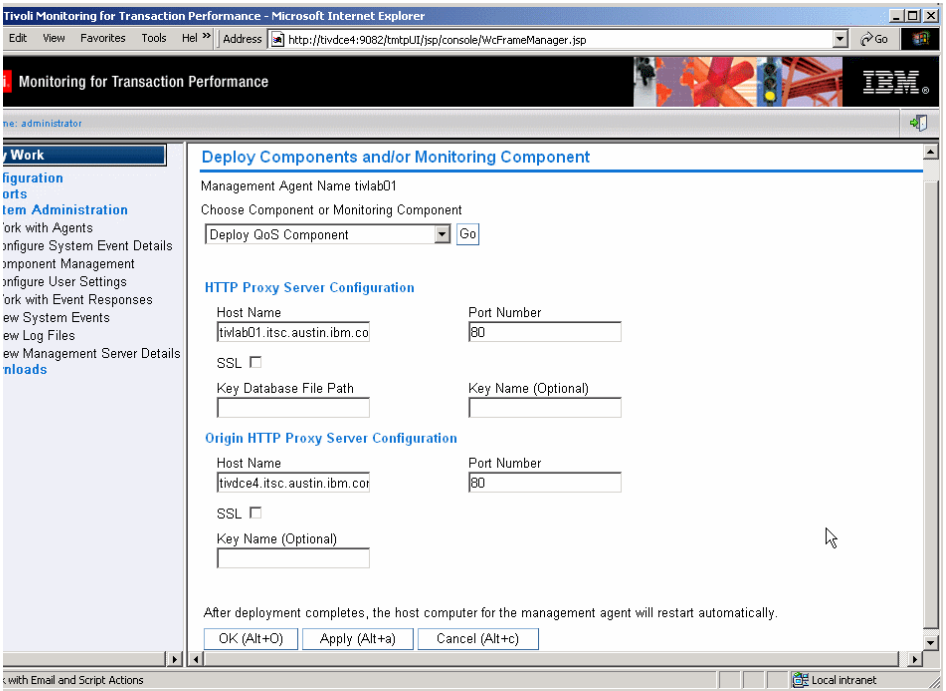


Figure 8-19 Deploy QoS components

The Deploy Components and/or Monitoring Component dialog shown in Figure 8-19 is used to configure the parameters for the QoS component. The information to be provided is grouped in two Server Configuration sections:

HTTP Proxy Specifies the networking parameters for the virtual server that will receive the requests for the origin server. The host name should be that of the Management Agent, which is the target of the QoS deployment, and the port number can be set to any free port on that system.

Origin HTTP Proxy

Specifies the networking parameters of the origin server, which will serve the requests forwarded from the virtual server residing on the QoS system. The host name should be set to the name of the system hosting the application server (for example, WebSphere Application Server), and the port number should be set to the port that the application server listens to for a particular application.

Provide the values as they apply to your environment, and click **OK** to start the deployment. After couple of minutes the Management Agent will be rebooted and the Quality of Service component has been deployed.

3. To verify that the installation was successful, refresh the Work with Agents dialog, and verify that the status for the QoS Component on the Management Agent in question shows Installed, as shown in Figure 8-20.

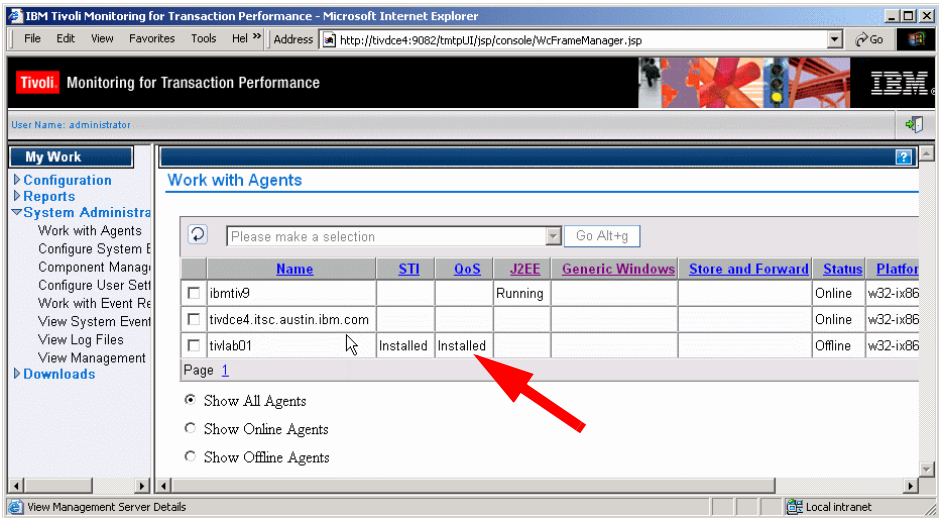


Figure 8-20 Work with Agents: QoS installed

8.5.2 Creating discovery policies for QoS

The purpose of the QoS discovery policy is to gather information about the URIs that are handled by the QoS Agent. As is the case for STI Agents, the URIs have to be discovered before monitoring policies can be defined and deployed. The Quality of Service discovery policy returns URIs only from Management Agents on which a Quality of Service listener is deployed.

Note: Please remember that specific discovery policies has to be created for each type of agent: QoS, J2EE, and STI.

Before setting up any policies for a QoS Agent, it is important to understand the concept of virtual servers.

The term *virtual server* refers to the practice of maintaining more than one server on one machine. These Web servers may be differentiated by IP, host name, and/or port number.

QoS and virtual servers

Even though the GUI for QoS configuration does not allow for defining multiple origin-server/virtual-server pairs, there is a way to use one QoS machine to measure requests for several back-end Web servers.

The advantage to this setup is that only one machine is used to measure the transactions' response times of a number of machines that do *real* work. However, one disadvantage of this setup is that the QoS system introduces a potential bottleneck and a single-point-of-failure. Another disadvantage is that there is no distinction in the metrics measured for the different servers, as the base for the distinguishing where the metrics come from is the QoS and not the back-end Web servers.

To set up a single QoS Agent to measure multiple back-end servers, please understand that because the QoS acts as a front end for the back-end Web server, the browsers connect to the QoS rather than to the Web server. If the QoS is to act as a front-end for different servers, it must have a separate identity for each server it serves as a front end for. To define separate identities, a virtual host has to be defined in the QoS HTTP server for each back-end server. These virtual servers may be either address- or name-based:

Address-based The QoS has multiple IP addresses and multiple network interfaces, each with its own host name.

Name-based The QoS has multiple host names pointing to the same IP address.

Both ways imply that the DNS server must be aware that the QoS has multiple identities.

Definitions of virtual servers are, after initial deployment of the Quality of Service component, performed by manually editing the HTTP configuration file on the QoS system. Example 8-2 shows an HTTP configuration file (http.conf) for a QoS system named tivlab01(9.3.5.14), which the alias of tivlab02(9.3.5.14), which is configured to use the default HTTP port (80). It has two virtual servers, backend1 and backend2, which in turn reverse proxy the hosts at 9.3.5.20 and 9.3.5.15.

Example 8-2 Virtual host configuration for QoS monitoring multiple application servers

```
# This is for name-based virtual host support.
NameVirtualHost backend1:80
NameVirtualHost backend2:80

# For clarity, place all listen directives here.
Listen 9.3.5.14:80
# This is the main virtual host created by install.       #
##########################################################
<VirtualHost backend1:80>
```

```
#SSLEnable
ServerName backend1

QoSMContactURL http://9.3.5.14:80/

# Enable the URL rewriting engine and proxy module without caching.
RewriteEngine on
RewriteLogLevel 0
ProxyRequests on
NoCache *

# Define a rewriting map with value-lists.
#             mapname key: filename
#RewriteMap    server
"txt:<QOSBASEDIR>/IBMHTTPServer/conf/apache-rproxy.conf-servers"

# Make sure the status page is handled locally and make sure no one uses our
# proxy except ourself.
RewriteRule    ^/apache-rproxy-status.*  -  [L]
RewriteRule    ^(https|http|ftp)://.*    -  [F]

# Now choose the possible servers for particular URL types.
RewriteRule    ^/(.*\.(cgi|shtml))$  to://9.3.5.20:80/$1  [S=1]
RewriteRule    ^/(.*)$               to://9.3.5.20:80/$1

# ... and delegate the generated URL by passing it through the proxy module
RewriteRule    ^to://([^/]+)/(.*)    http://$1/$2   [E=SERVER:$1,P,L]

# ... and make really sure all other stuff is forbidden when it should survive
# the above rules.
RewriteRule    .*                    -              [F]

# Setup URL reverse mapping for redirect reponses.
ProxyPassReverse    /    http://9.3.5.20:80/
ProxyPassReverse    /    http://9.3.5.20/
</VirtualHost>

#########################################################
# second backend machine created manually
#########################################################

<VirtualHost backend2:80>
#SSLEnable
ServerName backend2

QoSMContactURL http://9.3.5.14:80/

# Enable the URL rewriting engine and proxy module without caching.
RewriteEngine on
```

```
RewriteLogLevel 0
ProxyRequests on
NoCache *

# Define a rewriting map with value-lists.
#             mapname key: filename
#RewriteMap    server
"txt:<QOSBASEDIR>/IBMHTTPServer/conf/apache-rproxy.conf-servers"

# Make sure the status page is handled locally and make sure no one uses our
# proxy except ourself.
RewriteRule    ^/apache-rproxy-status.*  -  [L]
RewriteRule    ^(https|http|ftp)://.*    -  [F]

# Now choose the possible servers for particular URL types.
RewriteRule    ^/(.*\.(cgi|shtml))$  to://9.3.5.15:80/$1  [S=1]
RewriteRule    ^/(.*)$               to://9.3.5.15:80/$1

# ... and delegate the generated URL by passing it through the proxy module
RewriteRule    ^to://([^/]+)/(.*)    http://$1/$2   [E=SERVER:$1,P,L]

# ... and make really sure all other stuff is forbidden when it should survive
# the above rules.
RewriteRule    .*                    -              [F]

# Setup URL reverse mapping for redirect reponses.
ProxyPassReverse    /    http://9.3.5.15:80/
ProxyPassReverse    /    http://9.3.5.15/
</VirtualHost>
```

In a live production environment, chances are that multiple QoS systems will be used to monitor a variety of application servers hosting different applications, as depicted in Figure 8-21 on page 265.

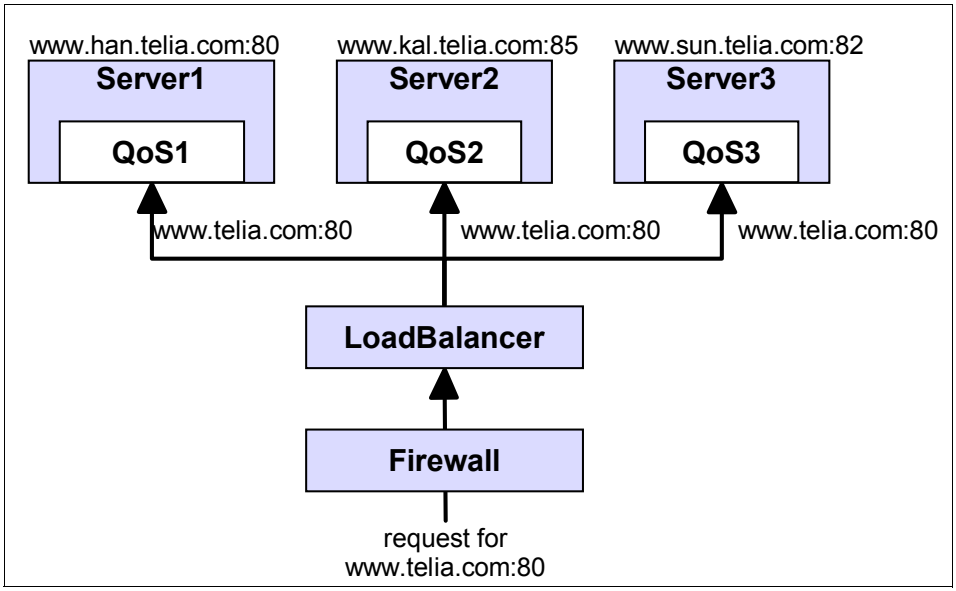


Figure 8-21 Multiple QoS systems measuring multiple sites

When planning to use multiple virtual servers on a single or multiple QoS system(s), please take the following into consideration:

Policy creation When scheduling a policy against particular end points, it makes sense to schedule it against groups that are created and maintained as virtual hosts. A customer that want to schedule a job against www.telia.com:80, for example, would want to select the group with all of the above QoS systems. When scheduling a policy against www.kal.telia.com:85, however, a group only contains QoS1. The name of the server QoS1 in this case does not give the user/customer any indication of what virtual hosts exist on each machine.

Endpoint Groups Endpoint Groups are an obvious match for this needed functionality. It is possible to name a group with the appropriate virtual host string (www.telia.com:80, for example).

Modification of Endpoint Groups for QoS Virtual Hosts

An extra flag will be added to the Object Model definition of an Endpoint Group to allow you to determine if each specific Endpoint Group is a virtual host. It will be a Boolean value for use by UI and the object model itself

Implications for UI The UI will need to only allow the scheduling of QoS policies against an Endpoint Group that is also a virtual host. The UI as well will need to not allow any editing/modification of Endpoint Groups that are virtual hosts; this will be handled by the QoS behavior on the Management Agents.

Update Mechanism Virtual hosts will be detected by the QoS component on each Management Agent. When the main QoS service is started on the Management Agent, a script will run, which will detect the virtual hosts installed on the particular Management Agent. Messages will then be sent to the Management Server; a Web service will be created on the Management Server as an interface to the session beans that will create, edit, and otherwise manage the endpoint groups that are virtual hosts.

Please consult the manual *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386 for more details.

Create discovery policies for QoS

Before creating a discovery policy for Quality of Service, you should note that QoS listening policies may be executed without prior discovery. However, if you do not know which areas of your Web environment require monitoring, create and run a discovery policy first and then create a listening policy.

To create a a QoS discovery policy for the home page of the TMTP Version 5.2 Console, select **Configuration** → **Work with Discovery Policies**. This will make the Work with Discovery Policies dialog shown in Figure 8-22 on page 267 appear.

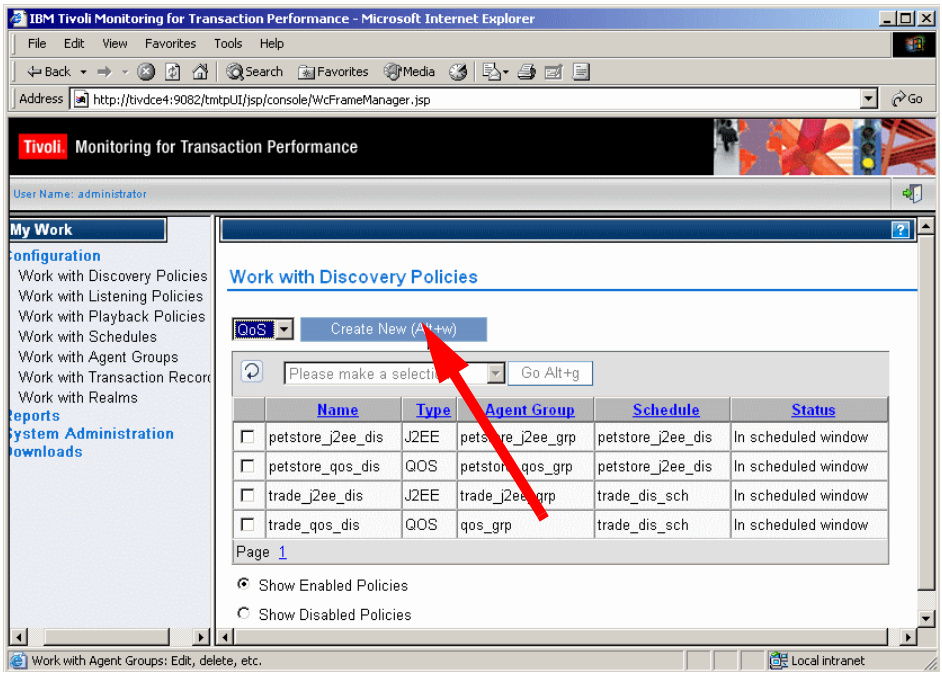


Figure 8-22 Work with discovery policies

To create a new policy, you should perform the following steps:

1. Select the QoS type of discovery policy, and click **Create New**, which will bring up the Configure QoS Listener dialog shown in Figure 8-23 on page 268.

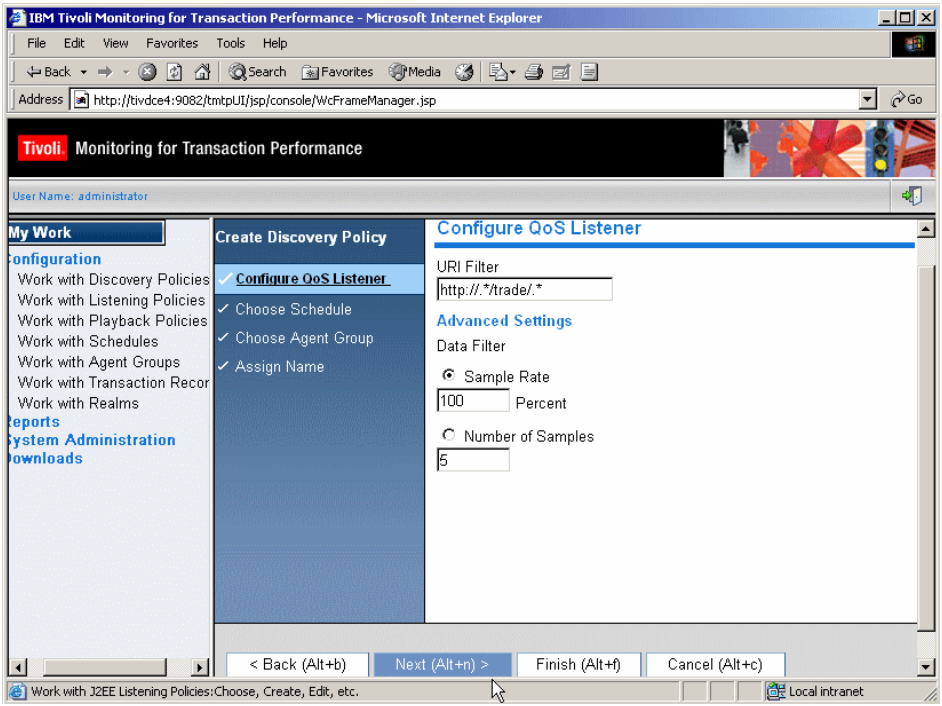


Figure 8-23 Configure QoS discovery policy

2. Add your URI filters and provide sampling information. Click **Next** to proceed to choose a schedule in the Work with Schedules dialog shown in Figure 8-24 on page 269.

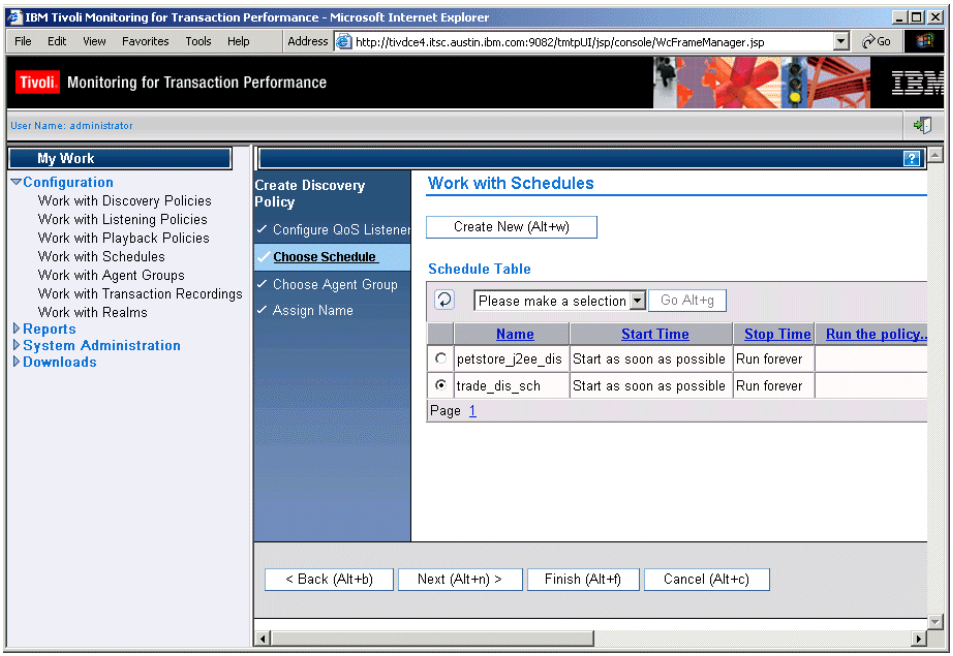


Figure 8-24 Choose schedule for QoS

3. Select a schedule, or create a new one that will suit your needs. Click **Next** to continue with Agent Group selection, as shown in Figure 8-25 on page 270.

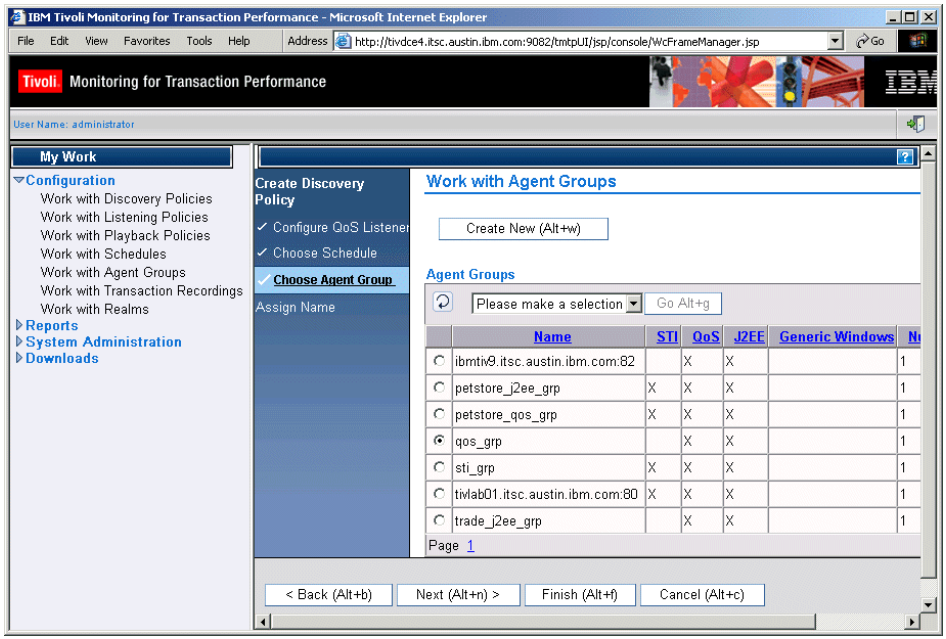


Figure 8-25 Selecting Agent Group for QoS discovery policy deployment

4. Before performing the final step, you have to select the group(s) of QoS Agents that the newly created QoS discovery policy will be distributed to. Select the appropriate group(s), and click **Next**.

5. Finally you have to provide a name. In this case *trade_qos-dis* is used. Also, determine if the profile is to be sent to the agents in the Agent Group(s) immediately, or wait until the next scheduled distribution. Click **Finish** to save the definition of the Quality of Service discovery profile (see Figure 8-26 on page 271).

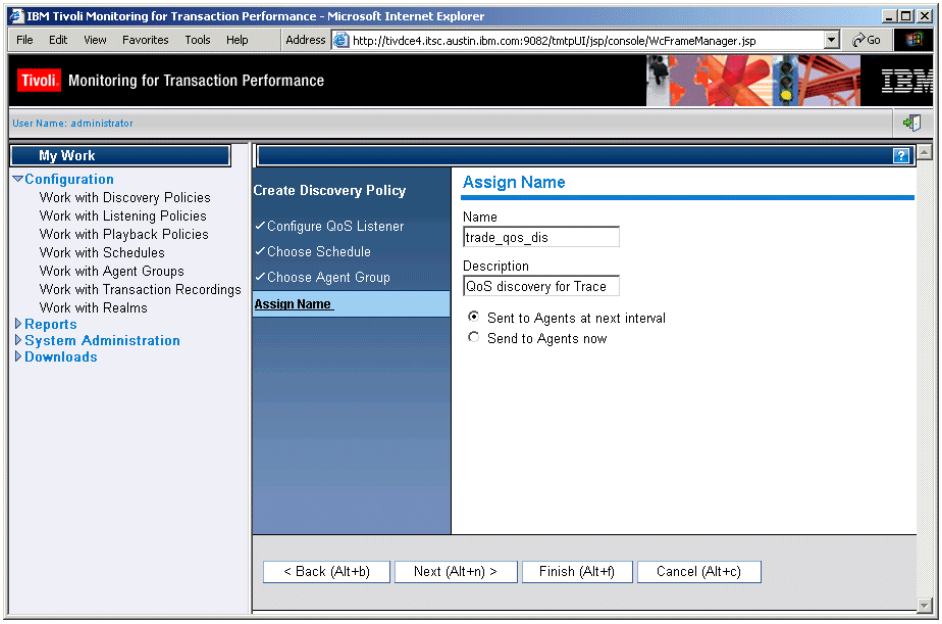


Figure 8-26 Assign name to new QoS discovery policy

Create a listening policy for QoS

The newly created discovery profile may be used as the starting point for creating the QoS listening policy (the one that actually collects and reports on transaction performance data). This will allow you to select transactions that have been discovered as the basis for the listening policy. Listening policies may also be created directly without the use of previously discovered transactions.

To create a listening policy by using the data gathered by the discovery policy, start by going to the home page of the TMTP Version 5.2 console and use the left side navigation pane to select **Configuration** → **Work with Discovery Policies**. The Work with Discovery Policies dialog shown in Figure 8-27 on page 272 will be displayed.

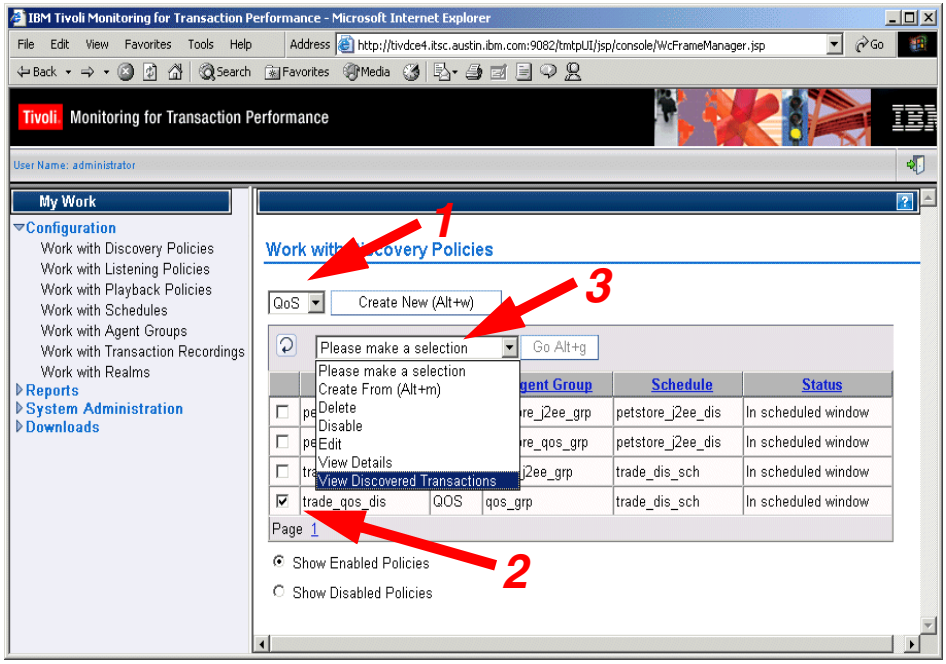


Figure 8-27 View discovered transactions to define QoS listening policy

Now, perform the following:

1. Select **QoS** and the desired type of policie(s) (QoS or J2EE) from the drop-down list at the top of the dialog.
2. Select the appropriate discovery policies. In our example, only trade_qos_dis was selected.
3. Select **View Discovered Transactions** from the drop-down list just above the list of discovery profiles and press **Go**. This will display a list of discovered transactions in the View Discovered Transactions, as shown in Figure 8-28 on page 273.

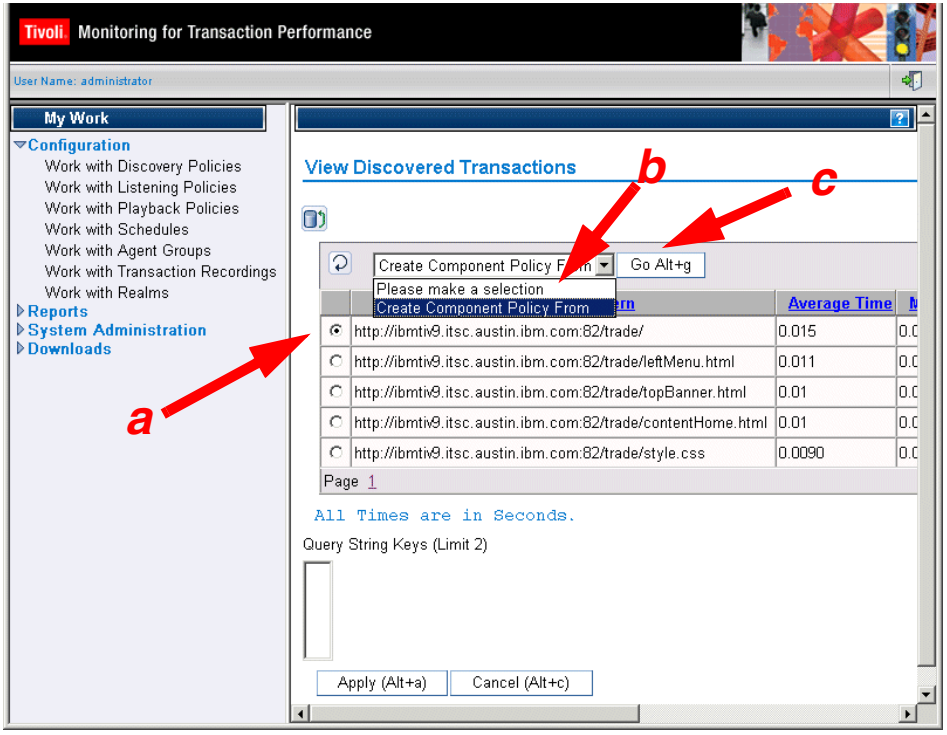


Figure 8-28 View discovered transaction of trade application

4. From the View Discovered Transactions dialog, select the transaction that will be the basis for the listening policy:

   a. Select a transaction.

   b. Select **Create Component Policy From** in the function drop-down menu at the top of the transaction list.

   c. Click **Go**.

   This will take you to the Configure QoS Listener dialog shown in Figure 8-29 on page 274.

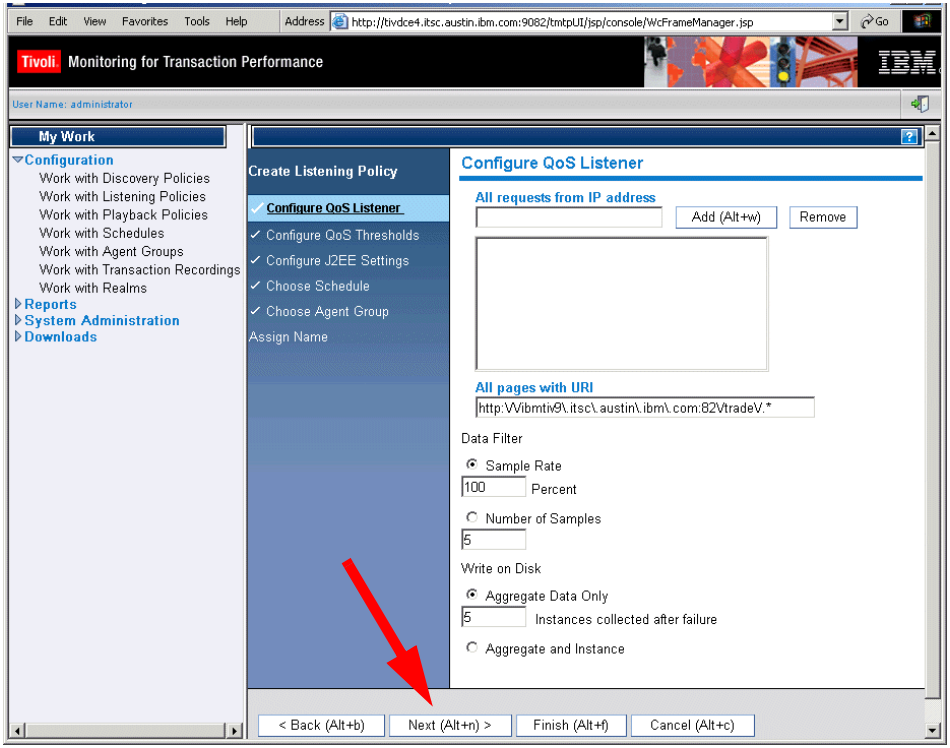


Figure 8-29 Configure QoS set data filter: write data

5. Apply appropriate values for filtering your data.

   You can apply filters that will help you collect transaction data from requests that originate from specific systems (IP addresses) or groups thereof. The filtering may be defined as a regular expression.

   In addition, you should specify how much data you want to capture per minute, and whether or not instance data should be stored along with the aggregated values. In case a threshold (which you will specify in the following dialog) is violated, TMTP Version 5.2 will automatically collect instance data for a number of invocations of the same transaction. You can customize this number to provide the level of detail needed in your particular circumstances.

   Click **Next** to go on to defining thresholds for the listening policy.

6. The Configure QoS Settings dialog, shown in Figure 8-30 on page 275, is used to define global values for threshold and event processing in QoS.

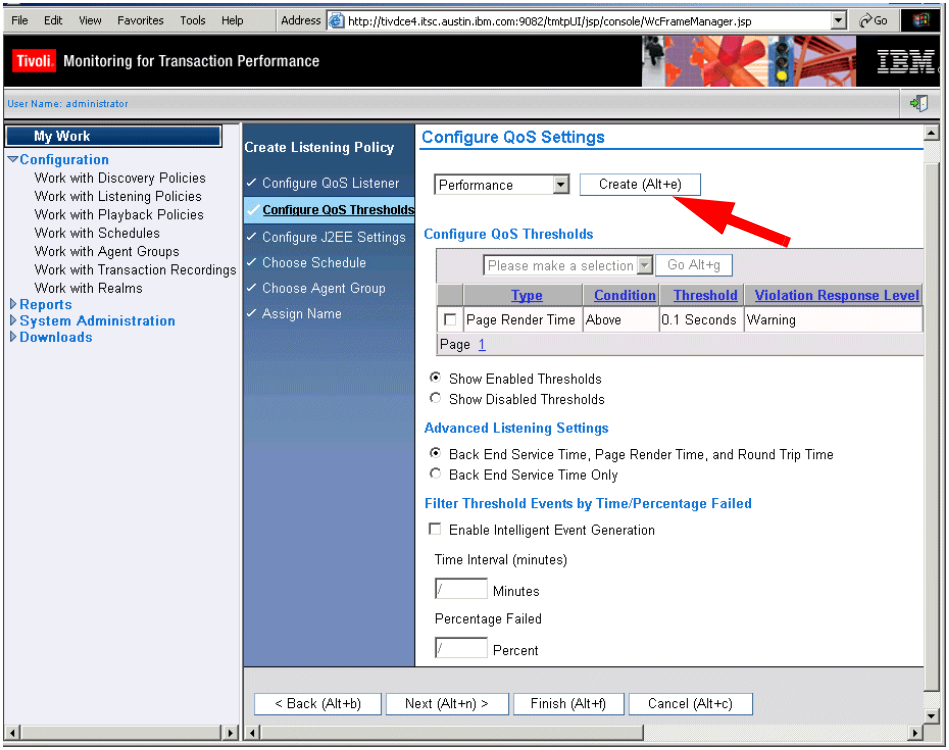


Figure 8-30 Configure QoS automatic threshold

To create a specific threshold, select the type in the drop-down menu under the dialog heading. Two types are available:

– Performance

– Transaction Status

When clicking **Create**, the Configure QoS Thresholds dialog shown in Figure 8-31 on page 276 will be displayed.

Detailed descriptions of each of the properties are available in the *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386.

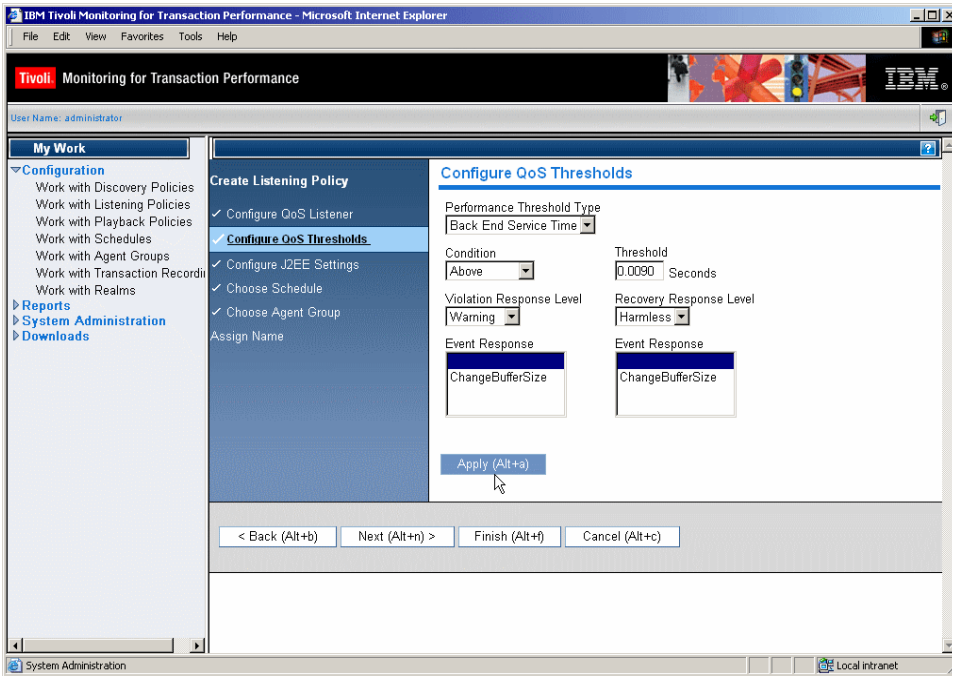


Figure 8-31 Configure QoS automatic threshold for Back-End Service Time

7. In the Configure QoS Thresholds, you can specify thresholds specific to each of the types chosen in the previous dialog.

   A Quality of Service transaction status threshold is used to detect a failure of the monitored transaction, or to detect the receipt of an HTTP response code from the Web server, or specific response times related to the QoS transaction during monitoring. Violation events are generated, or triggered, when failure occurs or when a specified HTTP response code is received. Recovery events and the associated notification are generated when the transaction executes as expected after a violation.

   Based on your selection, you can set thresholds for the following:

   Performance Back-End Service Time
   Page Render Time
   Round Trip Time

   Transaction Status Failure or specific HTTP return codes

   For each threshold you are creating, you should press **Apply** to save your settings, and when finished, click **Next** to continue to the Configure J2EE Settings dialog.

8. Since this does not provide functions for the QoS listening policy, click **Next** again to proceed to the schedule selection for the policy.
9. Schedules for Quality of Service listening policies are selected the same way as for any other policy. Please refer to 8.4.4, “Playback schedule definition” on page 248 for more details related to schedules. Click **Next** to go on to select Agent Groups for the listening policy.
10. Agent Group selection is common to all policy types. Please refer to the description provided in item 4 on page 270 for further details. Click **Next** to finalize your policy definition.
11. Having defined all the necessary properties of the QoS listening policy, all that is left before you can save and deploy the listening policy is to assign a name, and determine when to deploy the newly defined listening policy to the Management Agents.

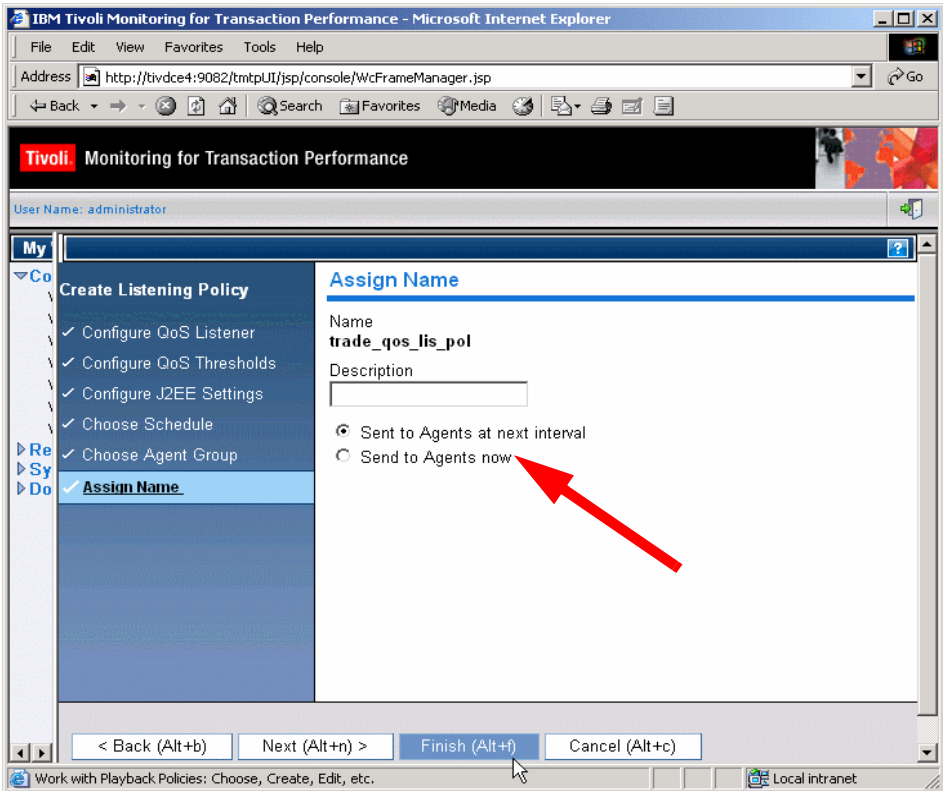


Figure 8-32 Configure QoS and assign name

From the Assign Name dialog shown in Figure 8-32, select your preferred distribution time and click **Finish.**

8.6 The J2EE component

The Java 2 Platform Enterprise Edition (J2EE) component of TMTP Version 5.2 is a component in IBM Tivoli Monitoring for Transaction Performance Version 5.2 that provides transaction decomposition capabilities for Java-based e-business applications.

Performance and availability information will be captured from methods of J2EE classes includes:

- Servlets
- Enterprise Java Beans (Entity EJBs and Session EJBs)
- JMS JDBC methods
- RMI-IIOP operations

The TMTP J2EE component supports WebSphere Application Server Enterprise Edition Versions 4.0.3 and later, which are valid for the J2EE monitoring component. Version 7.0.1 is the only supported version of BEA WebLogic.

More details about J2EE are available in 3.3.2, “J2EE instrumentation” on page 72.

8.6.1 J2EE component deployment

From a customization and deployment point of view the J2EE component is treated just like STI and QoS. A Management Agent can be instrumented to perform transaction performance measurements of this specific type of transactions, and it will report the findings back to the TMTP Management Server for further analysis and processing.

Use the following steps to deploy the J2EE component to an existing Management Agent:

1. Select **System Administration** → **Work with Agents** from the navigation pane on the TMTP console.
2. Select the Management Agent to which the component is going to be deployed, and choose **Deploy J2EE Monitoring Component** from the drop-down menu above the list of endpoints, as shown in Figure 8-33 on page 279. When ready, click **Go** to move on to configuring the specific properties for the deployment through the Deploy Components and/or Monitoring Component dialog, shown in Figure 8-34 on page 280.

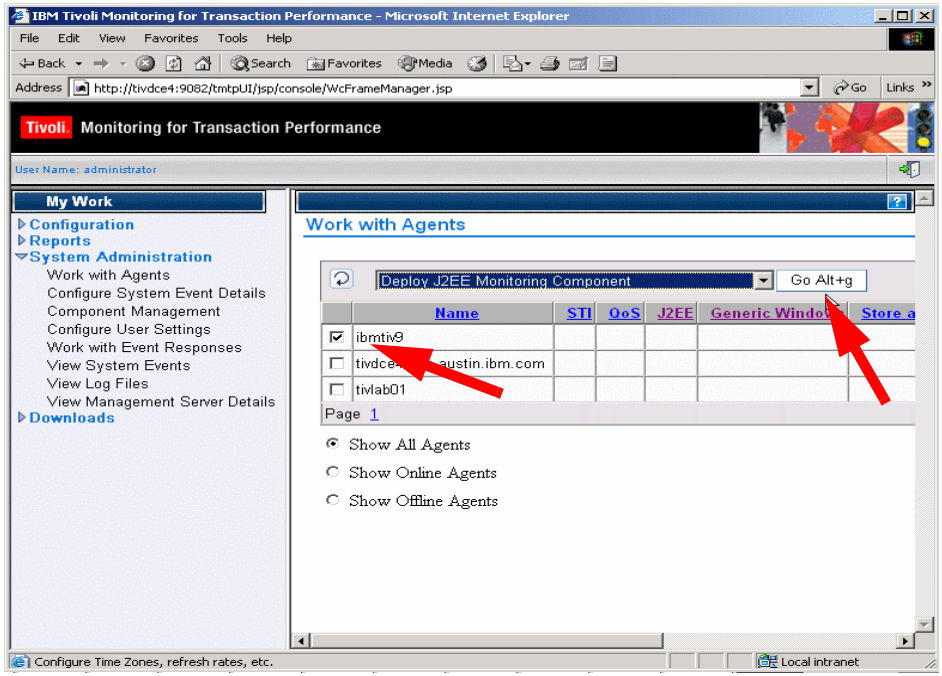


Figure 8-33 Deploy J2EE and Work of agents

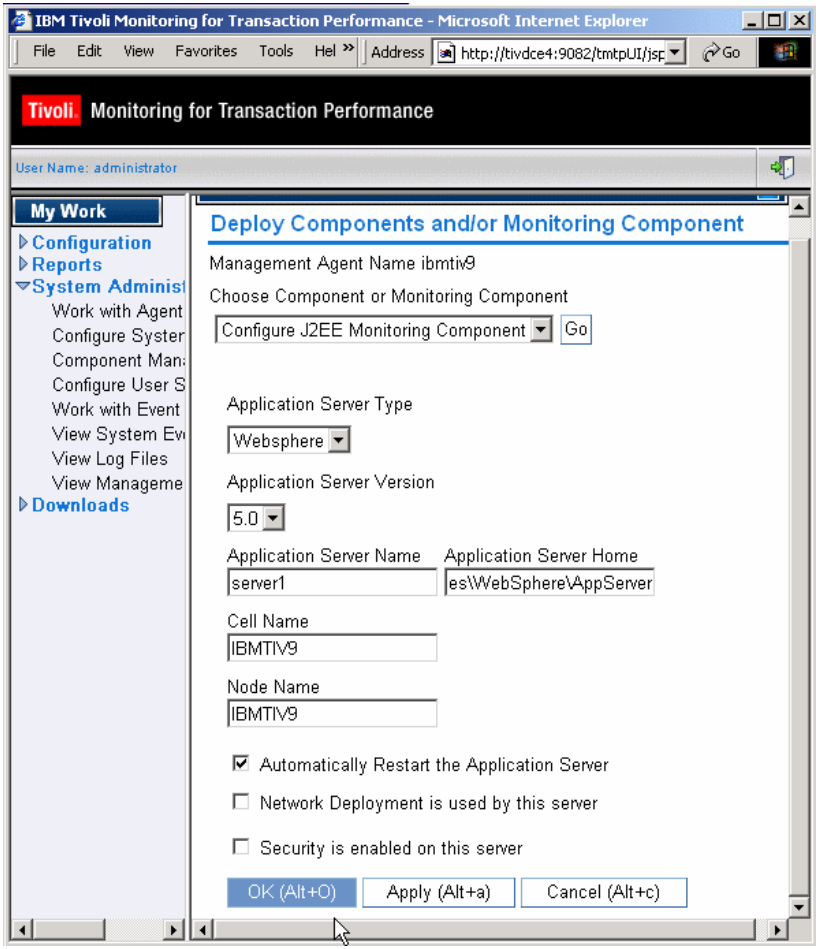


Figure 8-34 J2EE deployment and configuration for WAS 5.0.1

3. Select a specific make and model of application server that applies to your environment. The Deploy Components and/or Monitoring Component is built dynamically based upon the type of application server you select.

   The values you are requested to supply are summarized in Table 8-2 on page 281. Please consult the manual *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386 for more details on each of the properties.

Table 8-2 J2EE components configuration properties

Application Server make and model	Property	Example value
WebSphere Application Server Version 4	Application Server Name	*Default Server*
	Application Server Home	C:\WebSphere\AppServe
	Java Home	C:\WebSphere\AppServer\java
	Node Name	<YOUR MAs HOSTNAME>
	Administrative Port Number	8008
	Automatically Restart the Application Server	Check
WebSphere Application Server Version 5	Application Server Name	*server1*
	Application Server Home	C:\Progra~1\WebSphere\AppServer
	Java Home	C:\Progra~1\WebSphere\AppServer\java
	Cell Name	*ibmtiv9*
	Node Name	*ibmtiv9*
	Automatically Restart the Application Server	Check
Weblogic Version 7.0	Application Server Name	*petstoreServer*
	Application Server Home	c:\bea\weblogic700
	Domain	*petstore*
	Java Home	c:\bea\jdk131_03
	A script starts this server	Check in applicable
	Node Manager starts this server	Check in applicable

To define the properties for the deployment of the J2EE component to a Management Agent installed on a WebSphere Application Server 5.01 system, specify properties like the ones shown in Figure 8-34 on page 280 and click **OK** to start the deployment. After a couple of minutes, the Management Agent will be rebooted, and the J2EE component has been deployed.

4. To verify the success of the deployment, refresh the Work with Agents dialog, and verify that the status for the J2EE Component on the Management Agent in question shows Running, as shown in Figure 8-35.

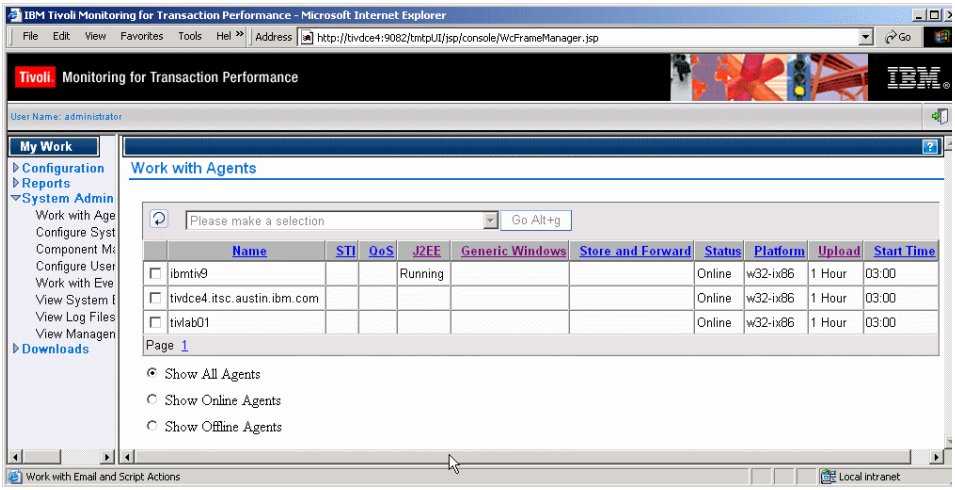


Figure 8-35 J2EE deployment and work with agents

8.6.2 J2EE component configuration

Once the J2EE component has been deployed, discovery and listening policies must be created and activated, as is the case for the other monitoring components: STI and QoS.

Creating discovery policies for J2EE

The J2EE discovery policies return URIs from Management Agents on which a J2EE listener is deployed. You might need to create more than one discovery policy to get a complete picture of an environment that includes both Quality of Service and J2EE listeners.

Please consult the manual *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386 for more details

The following outlines the procedure to create new discovery policies for a J2EE component:

1. Start by navigating to the Work with Discovery Policies dialog from the home page of the TMTP console. From the navigation pane on the left, select **Configuration** → **Work with Discovery Policies**.
2. In the Work with Discovery Policies dialog shown in Figure 8-36 on page 283, select a policy type of **J2EE** and press **Create New**.

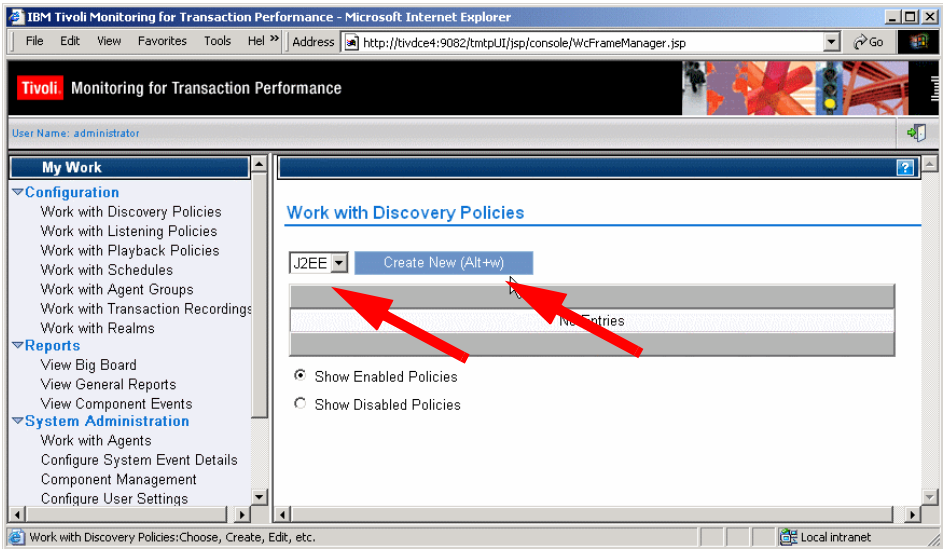


Figure 8-36 J2EE: Work with Discovery Policies

This will bring you to the Configure J2EE Listener dialog shown in Figure 8-37 on page 284, where you can specify filters and sampling properties for the J2EE discovery policy.

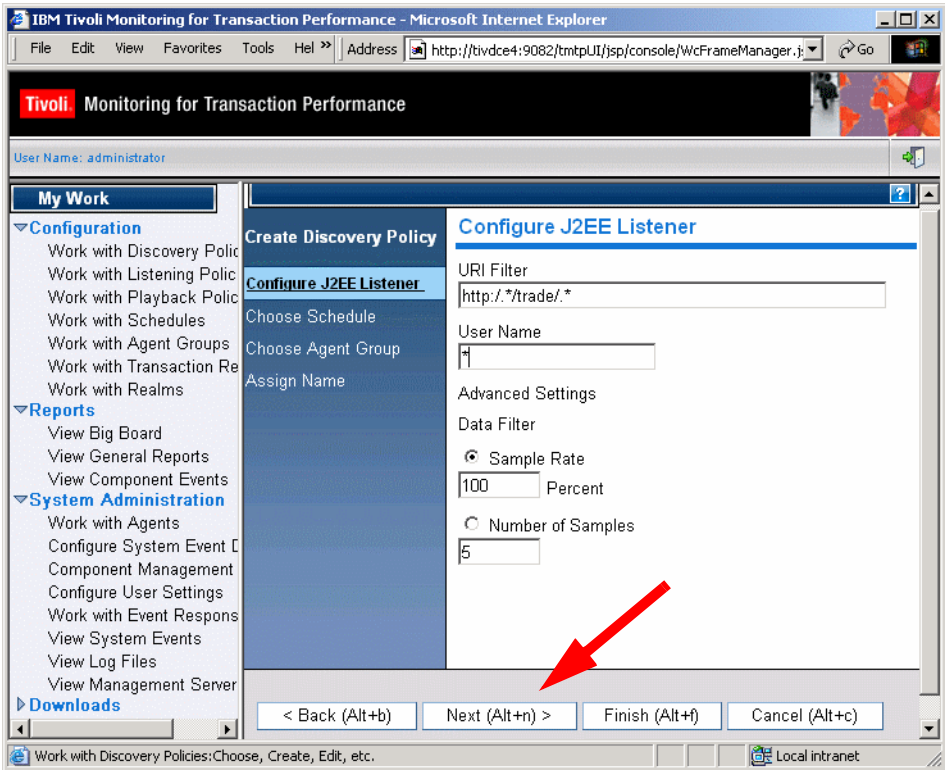


Figure 8-37 Configure J2EE discovery policy

3. Provide the filtering values of your choice, and click **Next** to proceed to schedule selection for the discovery policy.

   In the example shown in Figure 8-37, we want to discover all user requests to the trade application, as specified in the URI Filter and User name:

   URI Filter http://*/trade/*

   User name *

   Note: The syntax used to define filters are that of regular expressions. If your are not familiar with these, please refer to the appropiate appendix in the manual *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386.

4. Use the Work with Schedules dialog depicted in Figure 8-38 on page 285 to select a schedule for the discovery policy. Details regarding schedule definitions are provided in 8.4.4, “Playback schedule definition” on page 248.

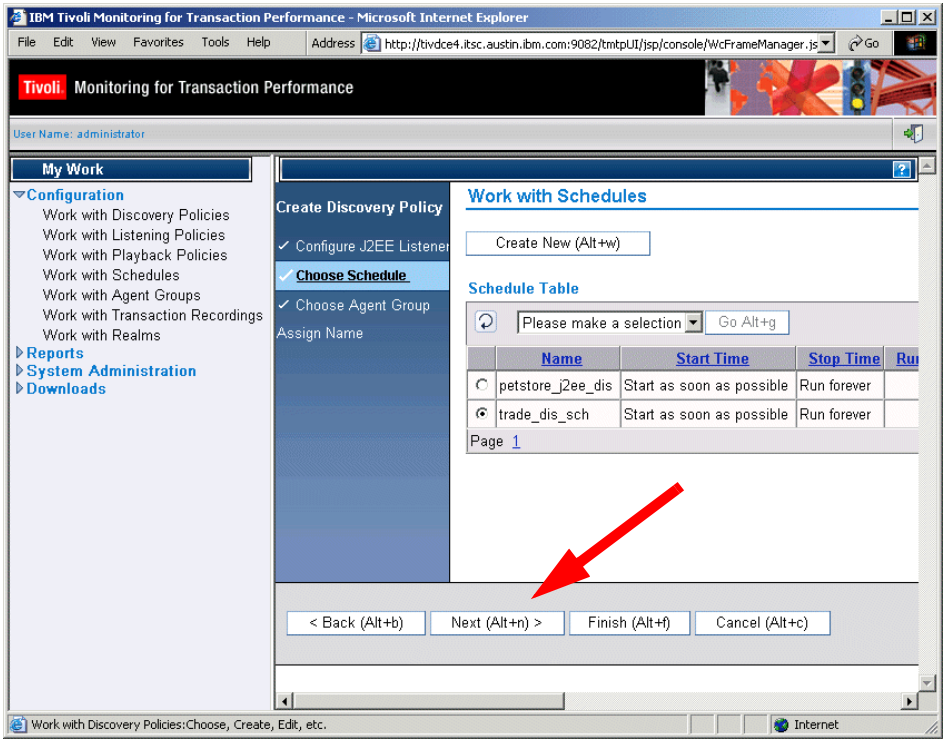


Figure 8-38 Work with Schedules for discovery policies

Click **Next** to select the target agents to which this policy will be distributed from the Agent Groups dialog.

5. Select the Agent Group(s) you wish to distribute the discovery policy to, and click **Next** to get to the final step in discovery policy creation: name assignment and deployment.

   In the example shown in Figure 8-39 on page 286, the group selected is named trade_j2ee_grp.

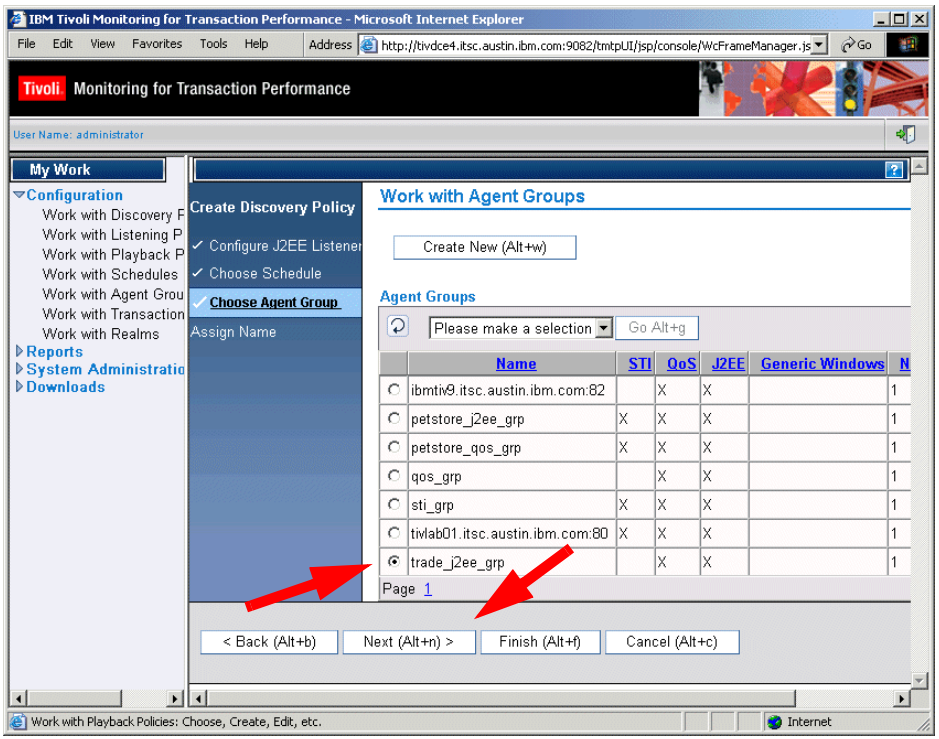


Figure 8-39 Assign Agent Groups to J2EE discovery policy

6. Assign a name to the new J2EE discovery policy, and determine when to deploy the policy. In the example shown in Figure 8-40 on page 287, the name assigned is trade_j2ee_dis, and it has been decided to deploy the policy at the next regular interval.

   Click **Finish** to complete the J2EE discovery policy creation.

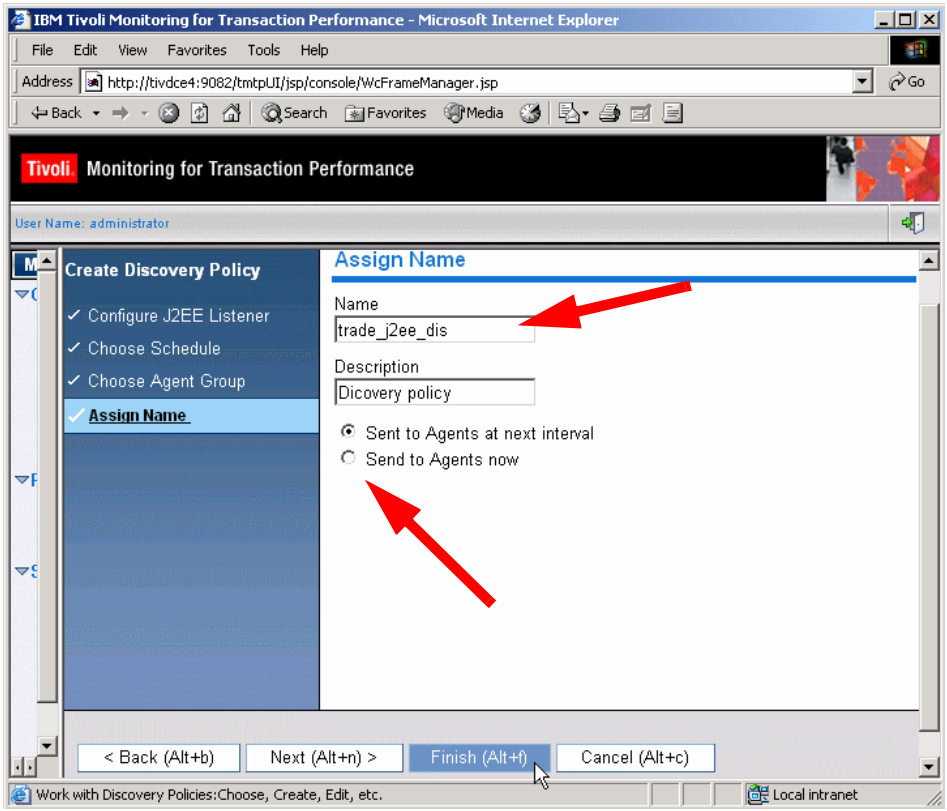


Figure 8-40 Assign name J2EE

In order to trigger the discovery policy, and to have transactions discovered, you need to direct your browser to the application and start a few transactions. In our example, we logged into the trade application at:

`http://ibmtiv9.itsc.austin.ibm.com/trade/app`

and started the Portfolio and Quotes/Trade transactions.

Creating J2EE listening policies

J2EE listening policies enable you to collect performance data for incoming transactions that run on one or more J2EE application servers. This will help you achieve the following:

- Measure transaction and subtransaction response times from J2EE applications in a real-time or simulated environment
- Perform detailed analysis of transaction performance data

- Identify root causes of performance problems

A J2EE listening policy instructs J2EE listeners that are deployed on Management Agents to collect performance data for transactions that run on one or more J2EE application servers. The Management Agents associated with a J2EE listening policy are installed on the J2EE application servers that you want to monitor. Running a J2EE listening policy produces information about transaction performance times and helps you identify problem areas in applications that are hosted by the J2EE application servers in your environment. A J2EE-monitored transaction calls subtransactions that are part of the transaction. There are six J2EE subtransaction types that you can monitor:

- Servlets
- Session beans
- Entity beans
- JMS
- JDBC
- RMI

When you create a J2EE listening policy, you specify a level of monitoring for each of the six subtransaction types. You also specify a range of other parameters to establish how and when the policy runs.

Perform the following steps to create J2EE listening policies:

1. Create and deploy a J2EE discovery policy, and make sure that the transactions you want to include in the listening policy have been discovered.
2. From the TMTP console home page, select **Configuration** → **Work with Discovery Policies** from the navigation pane on the left hand side. This will display the Work with Discovery Policies dialog, as shown in Figure 8-41 on page 289.

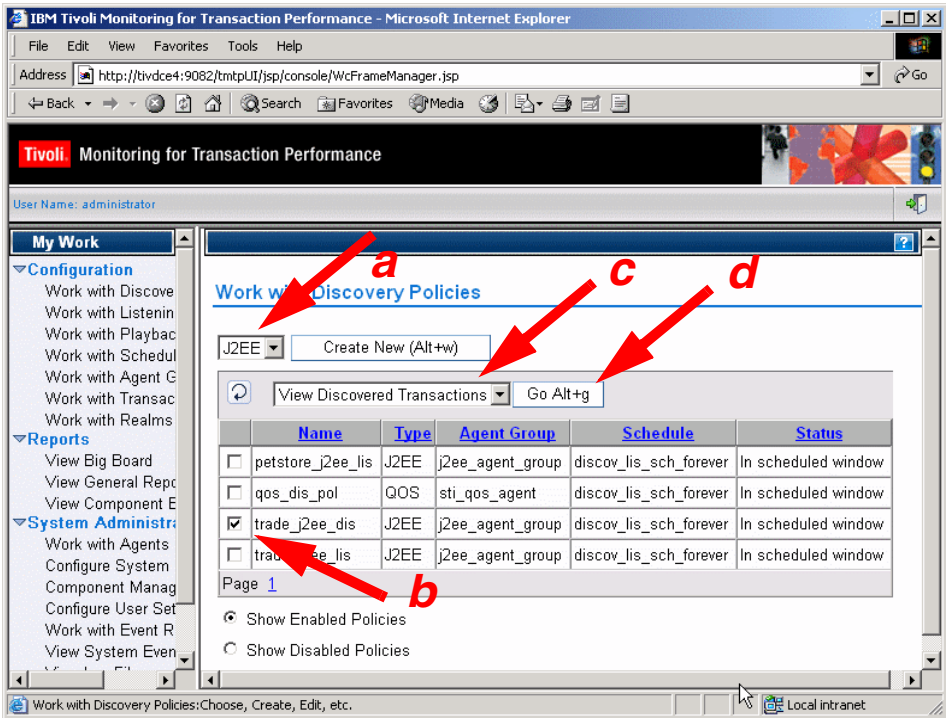


Figure 8-41 Create a listening policy for J2EE

3. Now, to choose the transactions to be monitored through this listening policy, perform the following:

   a. First, make sure that the active policy type is J2EE.

   b. Select the discovery policy of your interest.

   c. Select **View Discovered Transactions** from the action drop-down menu.

   d. Finally, click **Go** to open the View Discovered Transactions dialog, as depicted in Figure 8-42 on page 290.

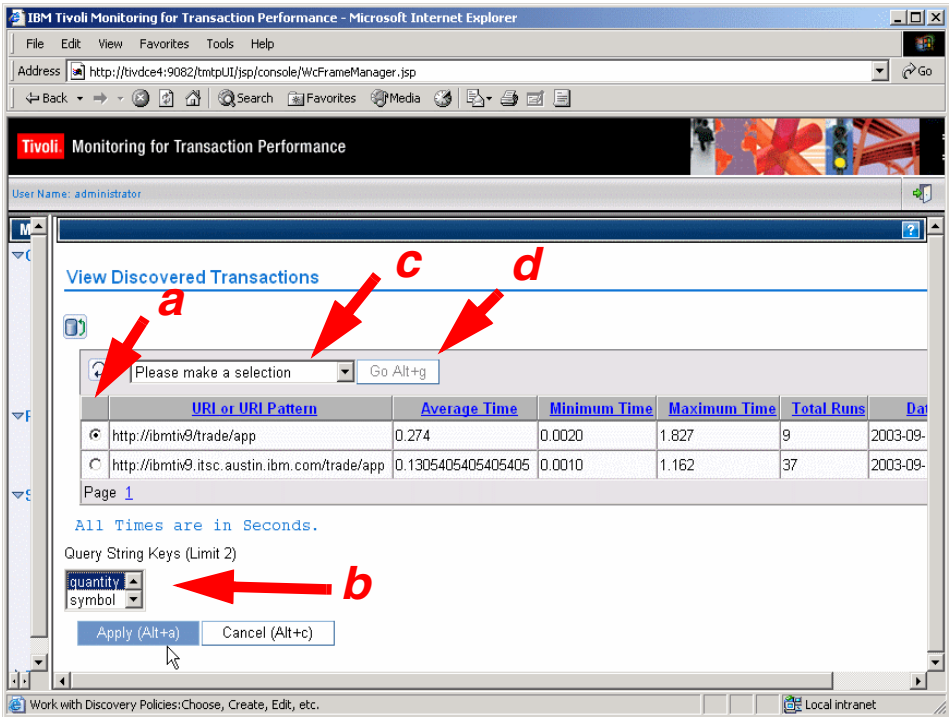


Figure 8-42 Creating listening policies and selecting application transactions

4. From the View Discovered Transactions, depicted in Figure 8-42, you select the specific transaction that you want to monitor. Now, perform the following:
   a. Make a selection for the **URI or URI Pattern** you want use to create listening policies.
   b. Select a maximum of two query strings for the listening policies, if any are available for the particular URI.
   c. Select **Create Component Policy From** in the action drop-down list.
   d. Press **Go**, and the Configure J2EE Listener dialog shown in Figure 8-43 on page 291 is displayed.

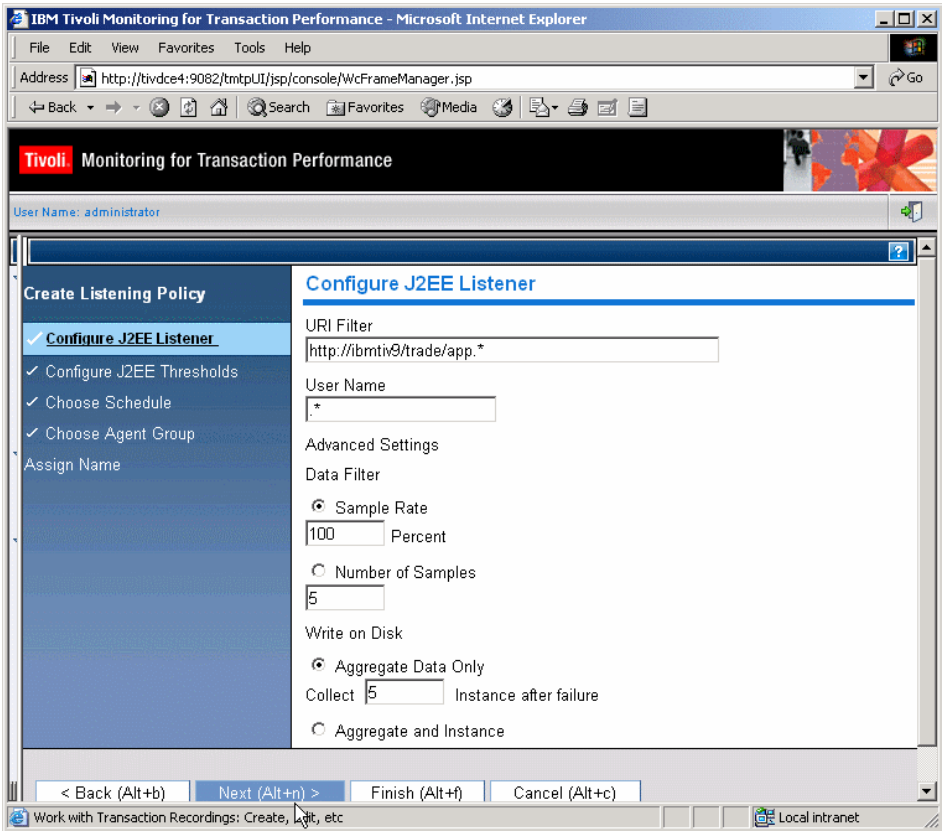


Figure 8-43 Configure J2EE listener

5. Choose the appropriate values for filtering and data collection and filtering.

   Selecting **Aggregate and Instance** specifies that both aggregate and instance data are collected. Aggregate data is an average of all of the response times detected by a policy. Aggregate data is collected at the monitoring agent once every minute. Instance data consists of response times that are collected every time the transaction is detected. All performance data, including instance and aggregate data, are uploaded to the Management Server once an hour by default. However, this value can be controlled through the Schedule Management Agent Upload dialog, which can be accessed from the TMTP console home page by navigating to **System Administration** → **Work with agent** → **Schedule a Collection**.

   For a high-traffic Web site, specifying **Aggregate and Instance** quickly generates a great deal of performance data. Therefore, when you use this option, specify a **Sample Rate** much lower than 100% or a relatively low **Number of Samples** to collect each minute.

6. Click **Next** to continue to the J2EE threshold definition, as shown in Figure 8-44.

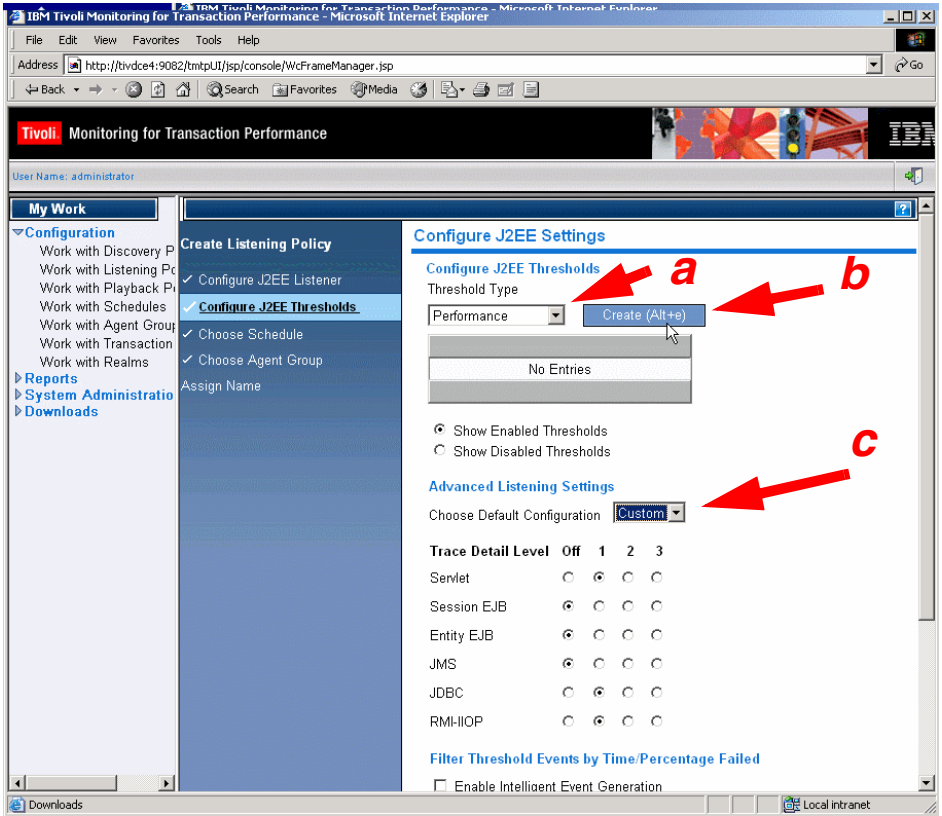


Figure 8-44 Configure J2EE parameter and threshold for performance

7. To set thresholds for event generation and problem identification for J2EE applications, do the following:

   a. Select the type of threshold you want to define. You may select between **Performance** and **Transaction Status**.

   b. Click **Create** to specify the transaction threshold details. These will be covered in detail in the following sections.

      You are not required to define J2EE thresholds in the current procedure. If you do, the thresholds apply to the transaction that is investigated, not to the J2EE subtransactions that are initiated by the transaction. After the policy runs, you can view a topology report, which graphically represents subtransaction performance and set thresholds on individual

subtransactions there. You can then edit the subtransaction thresholds in the current procedure.

c. Define the your J2EE trace configuration.

The J2EE monitoring component collects information for the servlet subtransaction type as follows. At trace level 1, performance data is collected, but no context information. At trace level 2, performance data is collected, along with some context information, such as the protocol that the servlet is using. At trace level 3, performance data and a greater amount of context information is collected, such as the ServletPath associated with the subtransaction.

Note: Under normal circumstances, specify a Low configuration. Only when you want to diagnose a performance problem should you increase the configuration to Medium or High.

If you specified a Custom configuration, you can adjust the level of monitoring for type-specific context information. Click one of the following radio buttons beside each of the J2EE subtransactions in the Trace Detail Level list:

Off	Specifies that no monitoring is to occur on the subtransaction.
1	Specifies that a low level of monitoring is to occur on the subtransaction.
2	Specifies that a medium level of monitoring is to occur on the subtransaction.
3	Specifies that a high level of monitoring is to occur on the subtransaction.

d. Define settings for intelligent event generation.

To enable intelligent event generation, perform the following actions in the Filter Threshold Events by Time/Percentage Failed fields:

i. Select the check box next to Enable Intelligent Event Generation.

While you are not required to enable intelligent event generation, do so in most cases. Without intelligent event generation, an overwhelming number of events can be generated. For example, a transaction might go above and fall below a threshold hundreds of times during a single monitoring period, and without intelligent event generation, each of these occurrences generates a separate event with associated notification. Intelligent event generation merges multiple threshold violations into a single event, making notification more useful and

reports, such as the Big Board and the View Component Events table, much more meaningful.

ii. Type 1, 2, 3, 4, or 5 in the Minutes field.

If you enable intelligent event generation, you must fill both the Minutes and the Percent Violations fields. The Minutes value specifies a time interval during which events that have occurred are merged. For example, if you specify two minutes, events are merged every two minutes during monitoring. Note that 1, 2, 3, 4, and 5 are the only allowed values for the Minutes field.

iii. Type a number in the Percent Violations field to indicate the percentage of transactions that must violate a threshold during the specified time interval before an event is generated.

For example, if you specify 80 in the Percent Violations field, 80% of transactions that are monitored during the specified interval must violate a threshold before an event is generated. The generated event describes the worst violation that occurred during the interval.

8. Schedules for J2EE listening policies are selected the same way as for any other policy. Please refer to 8.4.4, “Playback schedule definition” on page 248 for more details related to schedules. Click **Next** to go on to select Agent Groups for the listening policy.

9. Agent Group selection is common to all policy types. Please refer to the description provided in item 4 on page 270 for further details. Click **Next** to finalize your policy definition.

10. Having defined all the necessary properties of the J2EE listening policy, all that is left before you can save and deploy the listening policy is to assign a name, and determine when to deploy the newly defined listening policy to the Management Agents.

From the Assign Name dialog shown in Figure 8-45 on page 295 select your preferred distribution time, provide a name for the J2EE listening policy, and and click on **Finish**.

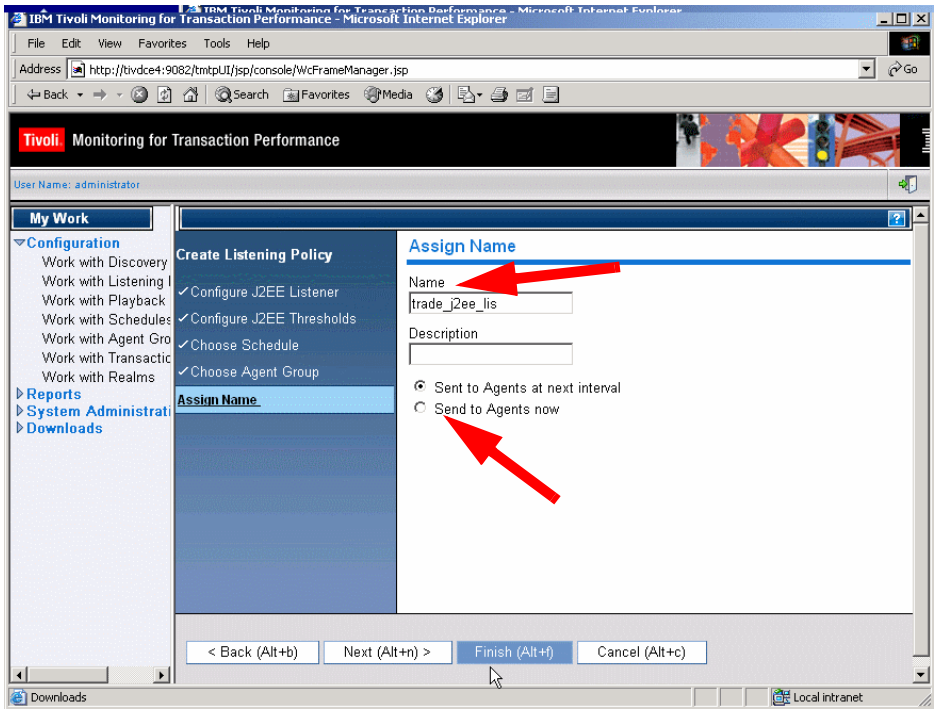


Figure 8-45 Assign a name for the J2EE listener

8.7 Transaction performance reporting

Before presenting the various online reports available with IBM Tivoli Monitoring for Transaction Performance Version 5.2 using the data from the sample Trade application, you should review the general description of online reporting in Chapter 7, “Real-time reporting” on page 211.

As a reminder, you should remember that IBM Tivoli Monitoring for Transaction Performance Version 5.2 provides three types of reports:

- Big boards
- General reports
- Components events

When working with the online reports, please keep the following in mind:

- All the online reports are available from the home page of the TMTP Console. Use the navigation pane on the left to go to Reports, and select the main category of our interest.

- To view the recent topology view for a specific QoS or J2EE enabled policy, go to the Big Board and click on the **Topology** icon of the transaction you are interested in.
- To view the most recent data, you can click on the **Retrieve Latest Data** icon (the hard disk symbol) in order to force the Management Agent to upload the latest data to the Management Server for storage in the TMTP database.
- In the topology views, you may change the filtering data type to **Aggregate** or **Instance** and **Show subtransaction slower than**.
- To see a general report of every transaction/subtransaction, select **General Reports** → **Transaction with Subtransaction** and use **Change Settings** to specify the particular policy for which you want see the details.
- To see the STI playback policy topology view, select **Topology** from the General Reports. Now use **Change setting** on the STI playback policy you want to see details for and drill down to the created view to see STI, QoS, and J2EE transaction correlation using ARM. For a discussion of transaction drill down using ARM and correlation, please see 7.4, “Topology Report overview” on page 215.
- There are four additional options from a topology node. Each of the following can be accessed using the context menu (right-click) of any object in the topology report:

  Events View View all the events for the policy and Management Agent.

  Response Time View View the node’s performance over time.

  Web Health Console Launch the ITM Web Health Console for this Management Agent.

  Thresholds View Configure a threshold for this node.

8.7.1 Reporting on Trade

If we consider an end user who uses a trade application for buying and selling stock, the application probably uses several processes to buy and sell, such as:

- Browse to Trade Web site
- Log in to trade application
- Quote/trade
- Buying/selling
- Log out from the application

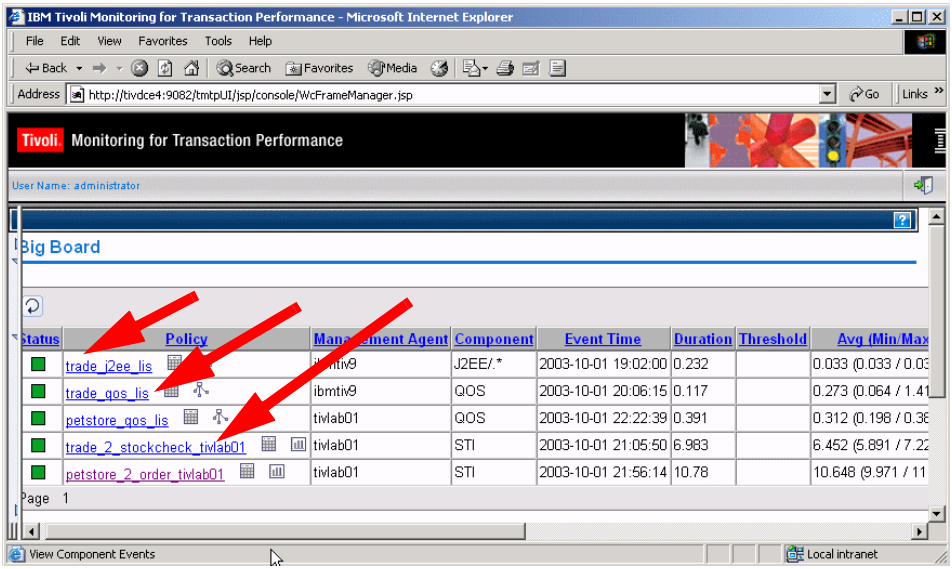

IBM Tivoli Monitoring for Transaction Performance - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://tivdce4:9082/tmtpUI/jsp/console/WcFrameManager.jsp
Go Links
Tivoli Monitoring for Transaction Performance
User Name: administrator
Big Board
Status Policy Management Agent Component Event Time Duration Threshold Avg (Min/Max
trade_j2ee_lis J2EE/.* 2003-10-01 19:02:00 0.232 0.033 (0.033 / 0.03
trade_qos_lis ibmtiv9 QOS 2003-10-01 20:06:15 0.117 0.273 (0.064 / 1.41
petstore_qos_lis tivlab01 QOS 2003-10-01 22:22:39 0.391 0.312 (0.198 / 0.38
trade_2_stockcheck_tivlab01 tivlab01 STI 2003-10-01 21:05:50 6.983 6.452 (5.891 / 7.22
petstore_2_order_tivlab01 tivlab01 STI 2003-10-01 21:56:14 10.78 10.648 (9.971 / 11
Page 1
View Component Events
Local intranet

Figure 8-46 Event Graph: Topology view for Trade application

The Trade application is running on a WebSphere Application Server Version 5.0.1 and we have configured a synthetic trade transaction with STI data to correlate J2EE components and Quality of Service, so we can figure out what is happening at the application server and database.

From the Big Board shown in Figure 8-46, we can see, because of our use of consistent naming standards, that the following active policies are related to the Trade application:

trade_j2ee_lis Listening policy for J2EE

trade_qos_lis Listening policy for QoS

trace_2_stock-check STI Playback policy

8.7.2 Looking at subtransactions

Now, to get a snapshot of the overall performance, we open the Transactions with Subtransactions report for the trace_2_stock-check policy. The overall and subtransaction times are depicted in Figure 8-47 on page 298.

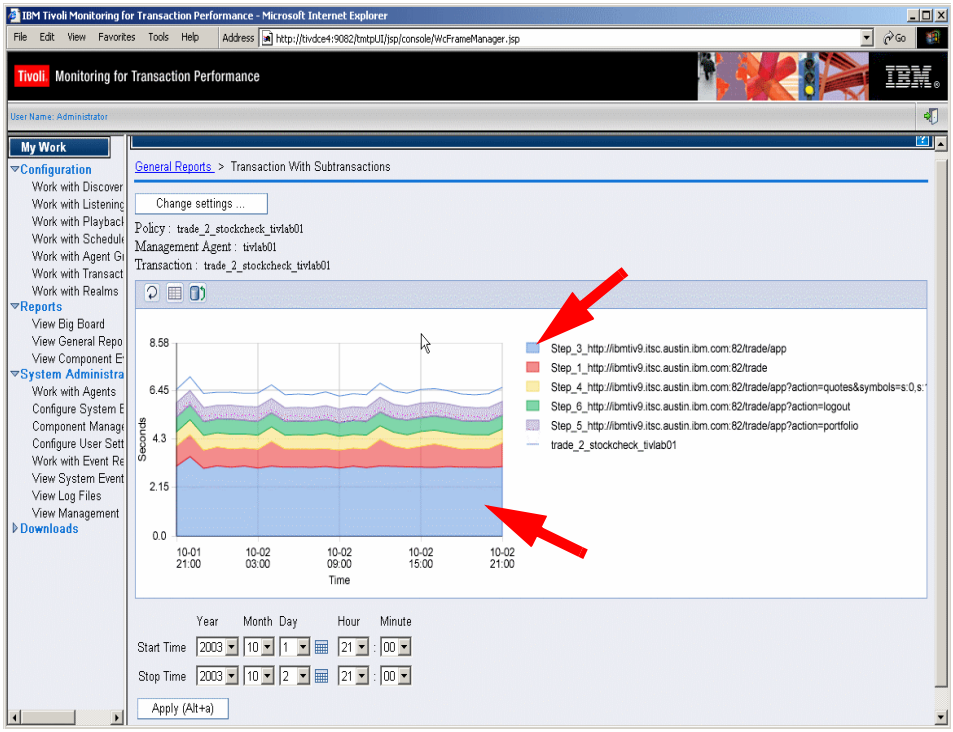


Figure 8-47 Trade transaction and subtransaction response time by STI

From the Transaction with Subtransaction report for the trace_2_stock-check, we see that the total User Experience Time to complete the order is 6.34 sec. This is measured by STI. We can drill down into the Trade application and see every subtransaction response time (maximum of five subtransactions) and understand how much time is used by every piece of the Trade business transaction.

Click on any subtransaction in the report, and it will drill down into the Back-End Service Time for the selected subtransaction. If this is repeated, TMTP will display the response times reported by the J2EE application components for the actual subtransaction. As an example, Figure 8-50 on page 301 shows the Back-End Service Time for the step_3 -- *app* -- subtransaction.

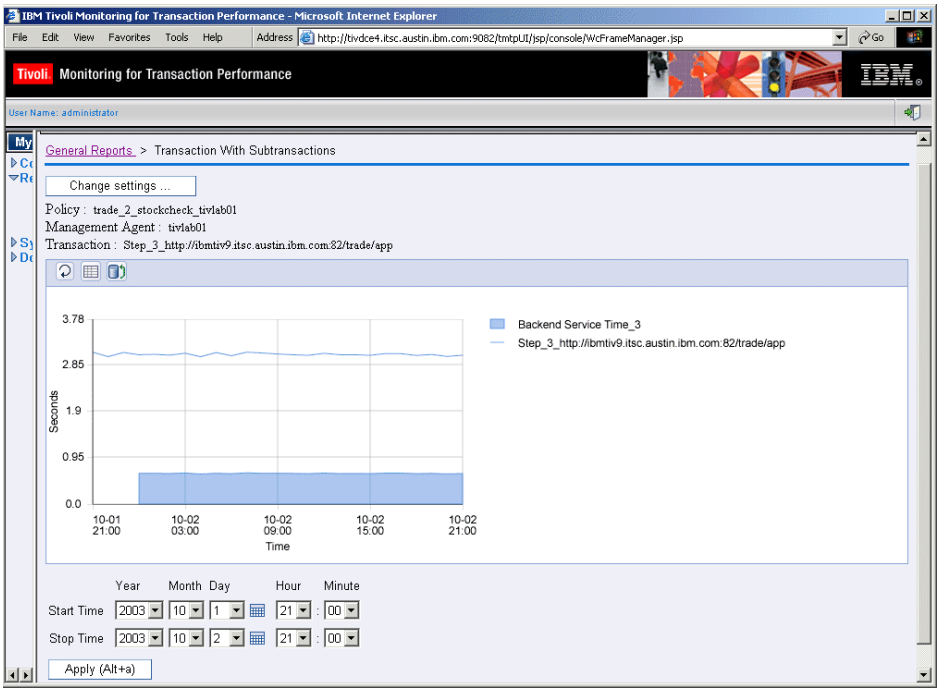


Figure 8-48 Back-End service Time for Trade subtransaction 3

The Back-End Service Time details for subtransaction 3 shows that the actual processing time was roughly one fourth of the overall time spent. When drilling further down into the Back-End Service TIme for subtransaction 3, we find, as shown in Figure 8-49 on page 300, that the servlet processing this request is:

```
com.ibm.websphere.samples.trade.web.OrdersAlertFilter.doFilter
```

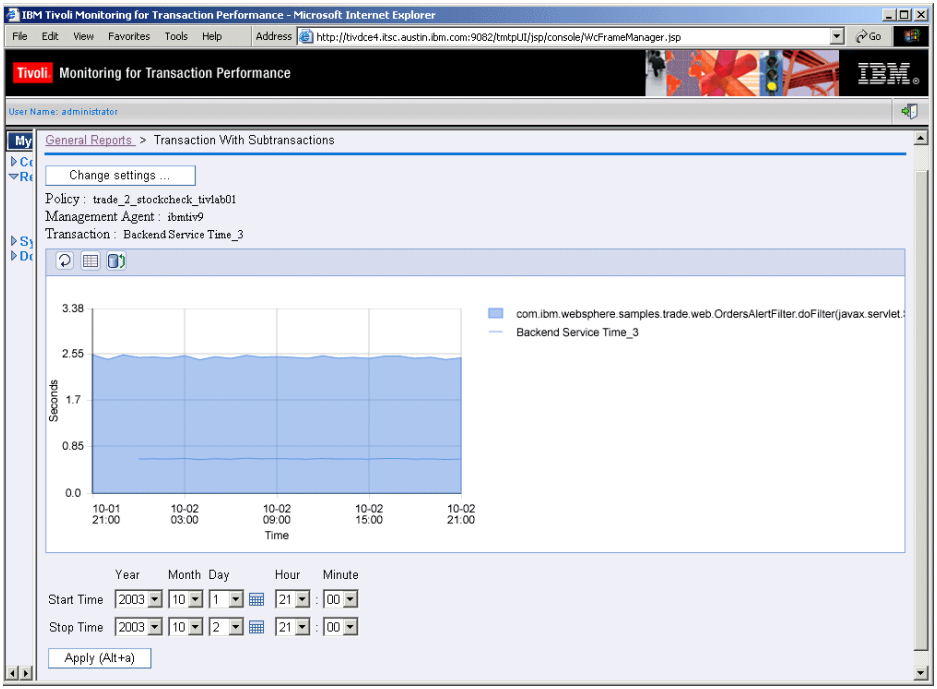


Figure 8-49 Time used by servlet to perform Trade back-end process

The drill down can basically go on and on until we have reached the lowest level in the subtransaction hierarchy.

8.7.3 Using topology reports

Another way of looking at the performance and responsiveness of the Trade application is to look at the topology. By drilling down into the QoS topology (by means of transactions and subtransactions, and using decomposing through relationships between parent- and child-transactions), we can find the real end-user response time, as shown in Figure 8-50 on page 301.

Because STI, QoS, and J2EE are ARM instrumented and parent/child relationships are correlated, we can also see these transactional relationships in the Topology View.

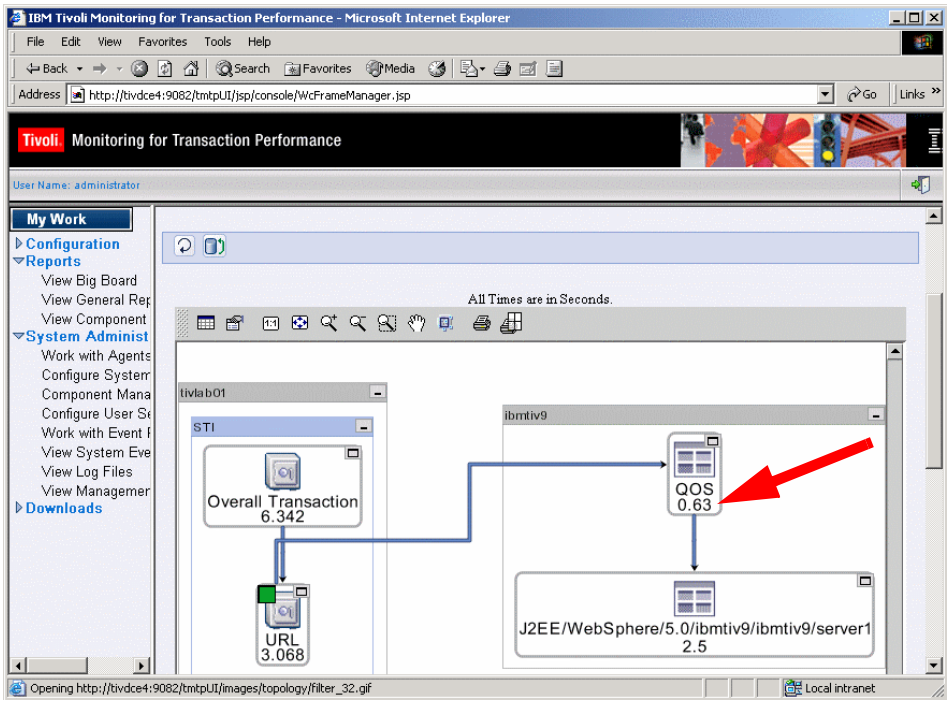


Figure 8-50 STI topology relationship with QoS and J2EE

The total real end-user response time is 0.623 seconds, and if we decompose the topology further, we see six specific back-end response times, one for each of the different Trade subtransactions/processes. From the Inspector View shown in Figure 8-51 on page 302, we can see the total end-user time, all subtransaction steps, Back-End Service Time, and J2EE application time from servlets, EJBs, and JSPs.

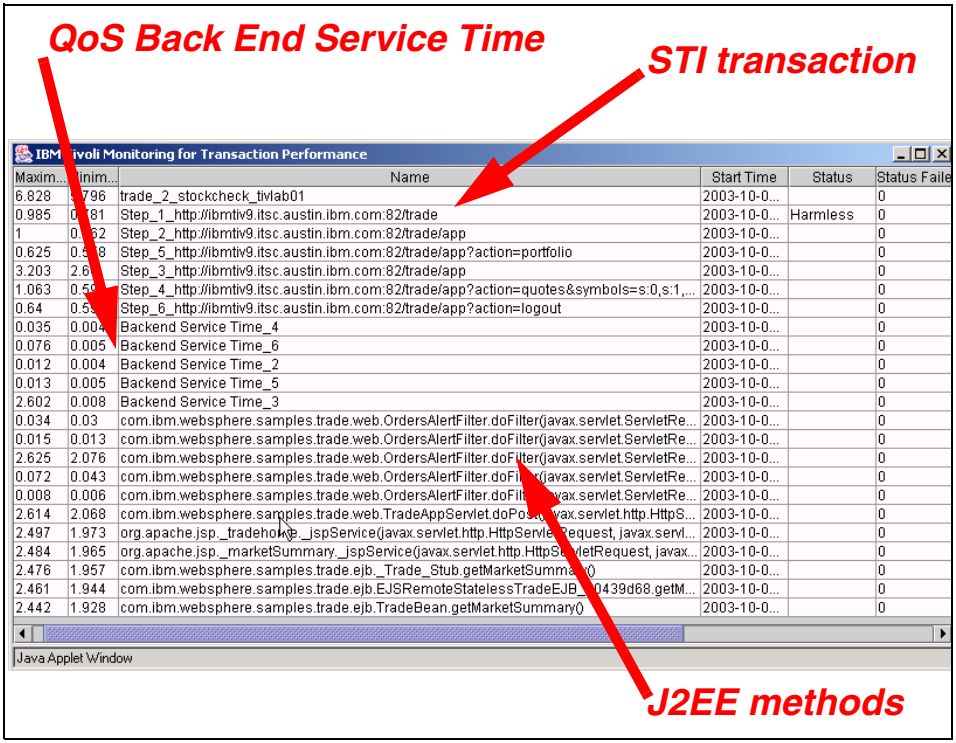

QoS Back End Service Time
STI transaction
IBM Tivoli Monitoring for Transaction Performance
Maxim...
Minim...
Name
Start Time
Status
Status Faile
6.828 0.796 trade_2_stockcheck_tivlab01 2003-10-0... 0
0.985 0.781 Step_1_http://ibmtiv9.itsc.austin.ibm.com:82/trade 2003-10-0... Harmless 0
1 0.962 Step_2_http://ibmtiv9.itsc.austin.ibm.com:82/trade/app 2003-10-0... 0
0.625 0.598 Step_5_http://ibmtiv9.itsc.austin.ibm.com:82/trade/app?action=portfolio 2003-10-0... 0
3.203 2.6 Step_3_http://ibmtiv9.itsc.austin.ibm.com:82/trade/app 2003-10-0... 0
1.063 0.59 Step_4_http://ibmtiv9.itsc.austin.ibm.com:82/trade/app?action=quotes&symbols=s:0,s:1,... 2003-10-0... 0
0.64 0.59 Step_6_http://ibmtiv9.itsc.austin.ibm.com:82/trade/app?action=logout 2003-10-0... 0
0.035 0.004 Backend Service Time_4 2003-10-0... 0
0.076 0.005 Backend Service Time_6 2003-10-0... 0
0.012 0.004 Backend Service Time_2 2003-10-0... 0
0.013 0.005 Backend Service Time_5 2003-10-0... 0
2.602 0.008 Backend Service Time_3 2003-10-0... 0
0.034 0.03 com.ibm.websphere.samples.trade.web.OrdersAlertFilter.doFilter(javax.servlet.ServletRe... 2003-10-0... 0
0.015 0.013 com.ibm.websphere.samples.trade.web.OrdersAlertFilter.doFilter(javax.servlet.ServletRe... 2003-10-0... 0
2.625 2.076 com.ibm.websphere.samples.trade.web.OrdersAlertFilter.doFilter(javax.servlet.ServletRe... 2003-10-0... 0
0.072 0.043 com.ibm.websphere.samples.trade.web.OrdersAlertFilter.doFi (javax.servlet.ServletRe... 2003-10-0... 0
0.008 0.006 com.ibm.websphere.samples.trade.web.OrdersAlertFilter.doFil vax.servlet.ServletRe... 2003-10-0... 0
2.614 2.068 com.ibm.websphere.samples.trade.web.TradeAppServlet.doPo vax.servlet.http.HttpS... 2003-10-0... 0
2.497 1.973 org.apache.jsp._tradehome._jspService(javax.servlet.http.HttpServle Request, javax.servl... 2003-10-0... 0
2.484 1.965 org.apache.jsp._marketSummary._jspService(javax.servlet.http.HttpS letRequest, javax... 2003-10-0... 0
2.476 1.957 com.ibm.websphere.samples.trade.ejb._Trade_Stub.getMarketSumma () 2003-10-0... 0
2.461 1.944 com.ibm.websphere.samples.trade.ejb.EJSRemoteStatelessTradeEJB_ 0439d68.getM... 2003-10-0... 0
2.442 1.928 com.ibm.websphere.samples.trade.ejb.TradeBean.getMarketSummary() 2003-10-0... 0
Java Applet Window
J2EE methods

Figure 8-51 QoS Inspector View from topology correlation with STI and J2EE

However, so far, we have not analyzed how much time is spent in the WebSphere Application Server 5.0.1 application server and database, that is, the combined total for:

- Trade EJB
- Trade session EJB
- Trade JSP pages
- Trade JavaServlet
- Trade JDBC
- Trade database

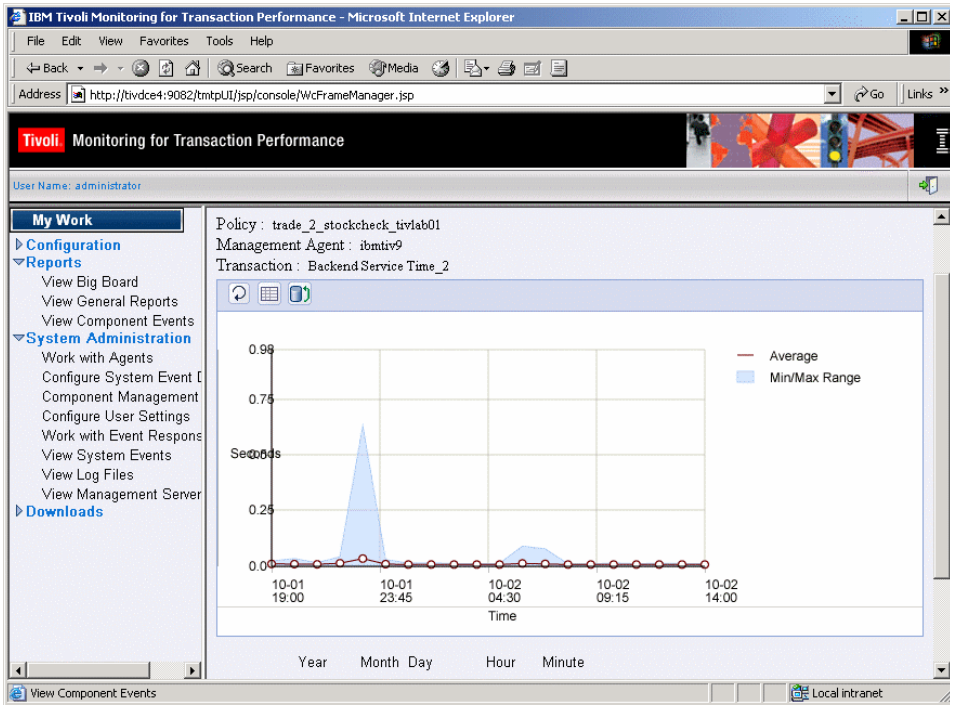


Figure 8-52 Response time view of QoS Back end service(1) time

Looking at the overall Trade application response time (shown in Figure 8-52), we can break down the application response time:

- EJB response time (see Figure 8-53 on page 304 and Figure 8-54 on page 305)
- JSPpages response time
- JDBC response time (see Figure 8-55 on page 306)

and drill down to its child methods or execution. In this way, we can find any bottleneck of the application server, database, or HTTP server by using different TMTP components, synthetic and real.

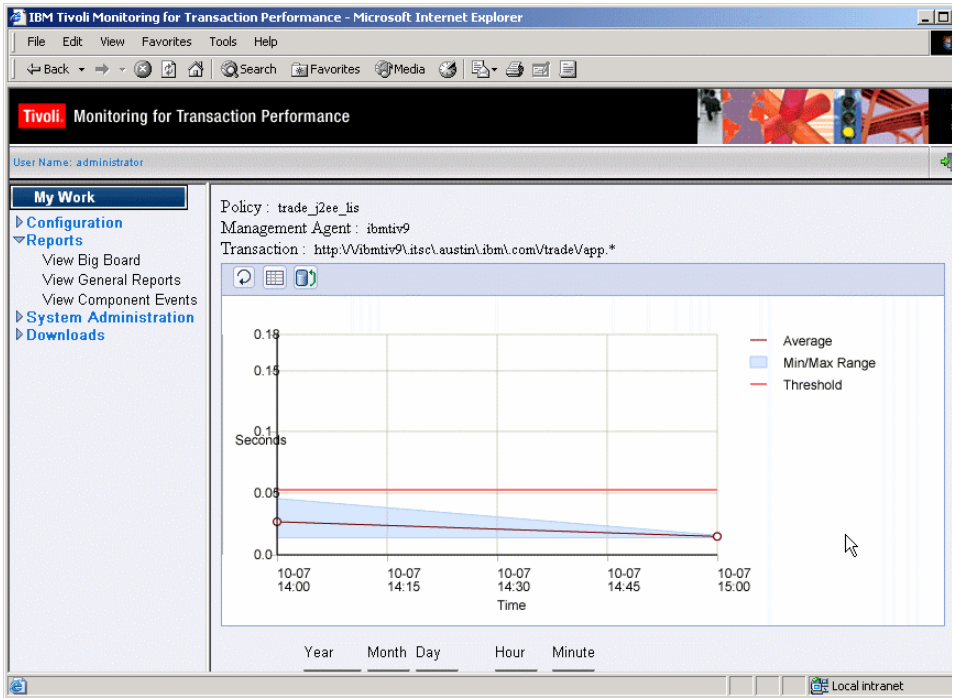


Figure 8-53 Response time view of Trade application relative to threshold

Figure 8-53 shows the overall Trade application response time relative to the defined threshold instead of the absolute times shown in Figure 8-52 on page 303.

When drilling down into the Trade application response times shown in Figure 8-53, we see the response times form the getMarketSummery() EJB (see Figure 8-54 on page 305).

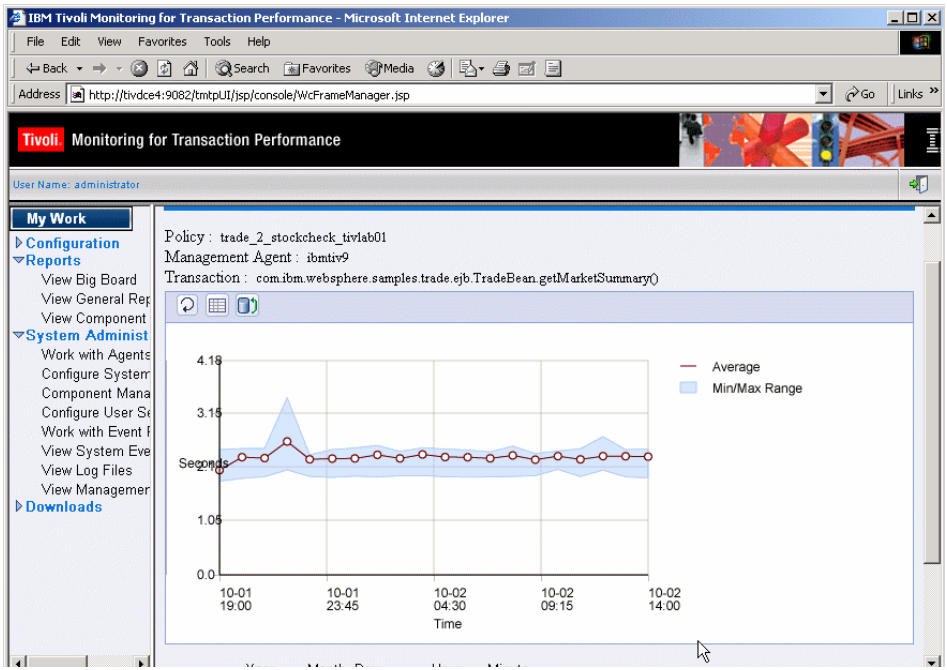


Figure 8-54 Trade EJB response time view get market summary()

Figure 8-55 on page 306 shows you how to drill all the way into a JDBC call to identify the database related bottlenecks on a per-statement basis.

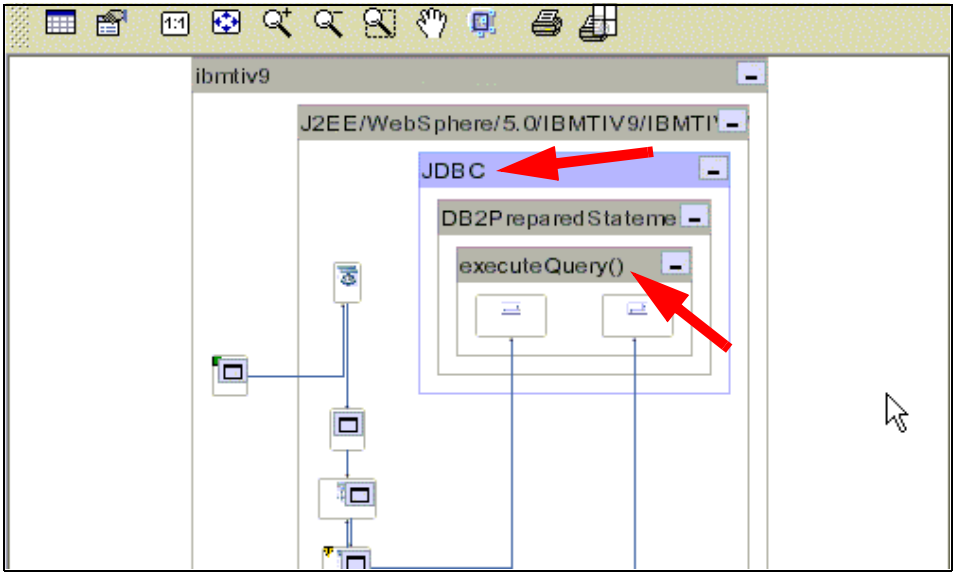


Figure 8-55 Topology view of J2EE and trade JDBC components

For root cause analysis, we can combine the topology view (showing the e-business transaction/subtransaction and EJB, JDBC, and JSP methods with ITM events of different resource models like CPU, processor, database, Web, and Web application using the ITM Web Health Console. Ultimately, we can send the violation event to TEC. Figure 8-56 on page 307 shows you how to launch the ITM Health Console directly from the topology view.

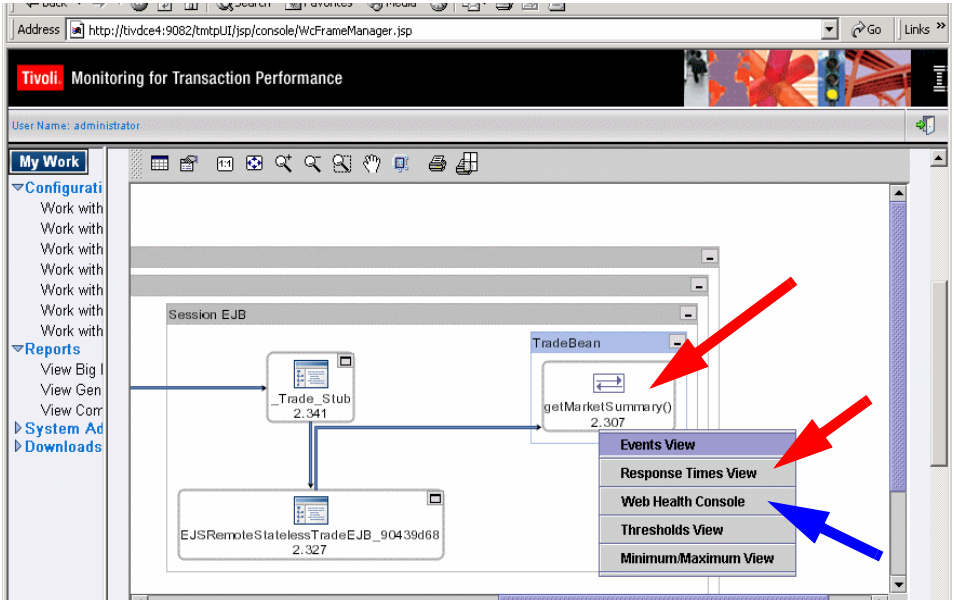


Figure 8-56 Topology view of J2EE details Trade EJB: get market summary()

8.8 Using TMTP with BEA Weblogic

This section discusses how to implement and configure the J2EE components in a BEA Weblogic application server environment.

In this section, we introduce the Pet Store sample business application and demonstrate drill down into all the business processes step by step. In addition, front-end as well as back-end reports are provided for all activities, in order to illustrate how IBM Tivoli Monitoring for Transaction Performance Version 5.2 standard components can be applied to a Weblogic environment to:

- Measure real-time Web transaction performance
- Measure synthetic end-user time
- Identify bottlenecks in the e-business processes

This section contains the following:

- 8.8.1, “The Java Pet Store sample application” on page 308
- 8.8.2, “Deploying TMTP components in a Weblogic environment” on page 310
- 8.8.3, “J2EE discovery and listening policies for Weblogic Pet Store” on page 312

- 8.8.4, “Event analysis and online reports for Pet Store” on page 316

8.8.1 The Java Pet Store sample application

The WebLogic Java Pet Store application is based on the Sun Microsystems Java Pet Store 1.3 demo. The Java Pet Store 1.3 is a J2EE sample application. It uses a combination of Java and J2EE technologies, including:

- The JavaServer Pages (JSP) technology
- Java servlets, including filters and listeners
- The Java Message Service (JMS)
- Enterprise JavaBeans, including Container Managed Persistence (CMP), Message Driven Beans (MDB), and the EJB Query Language (EJB QL).
- A rich client interface built with the Java Foundation Classes (JFC) and Swing GUI components
- XML and Extensible Style Sheets for Transformation (XSLT), a reusable Web application framework.

The welcome dialog is provided in the window shown in Figure 8-57 on page 309, and technical details are available at:

http://java.sun.com/features/2001/12/petstore13.html

Figure 8-57 Pet Store application welcome page

The Pet Store application uses a PointBase database for storing data. It will populate all demonstration data automatically when an application is run for the first time.

Once installed, you can log in to Weblogic Administration console (see Figure 8-58 on page 310) to see details for the Pet Store application components and configuration.

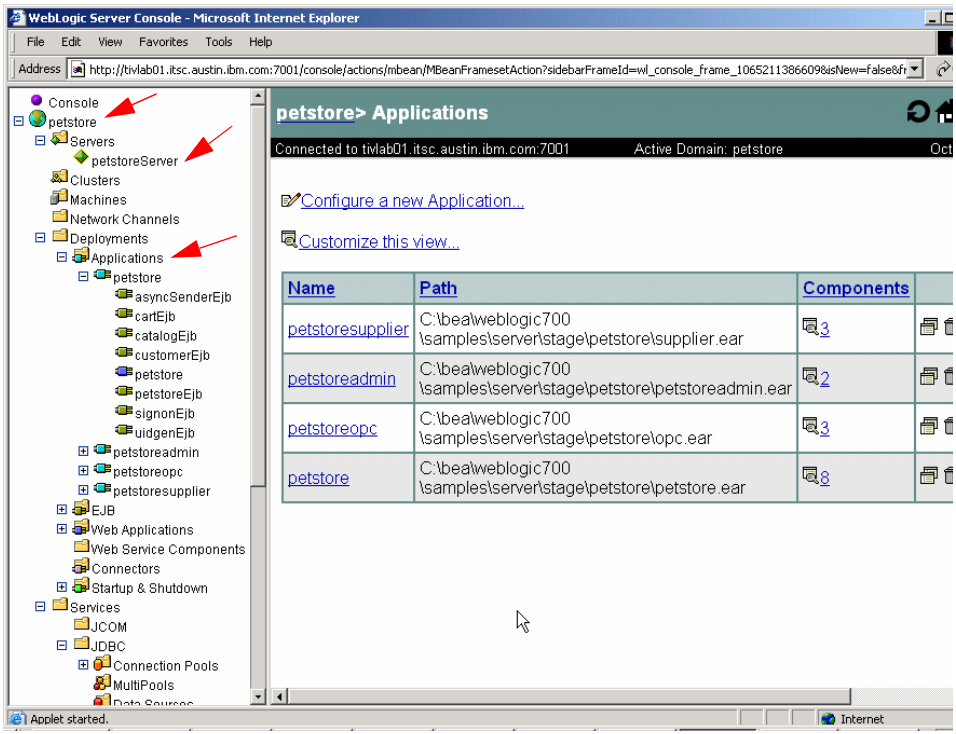


Figure 8-58 Weblogic 7.0.1 Admin Console

To start the Pet Store application from the Windows Desktop, select **Start** → **Programs** → **BEA Weblogic Platform 7.0** → **Weblogic Server 7.0** → **Server Tour and Examples** → **Lunch Pet Store**.

8.8.2 Deploying TMTP components in a Weblogic environment

The deployment of the IBM Tivoli Monitoring for Transaction Performance Version 5.2 Management Agents and monitoring components is similar to the procedures already described for deployment and configuration in a WebSphere Application Server environment. Please refer to the following sections for the specific tasks.

- 4.1.4, “Installation of the Management Agents” on page 130
- 8.4, “STI recording and playback” on page 241
- 8.5, “Quality of Service” on page 257
- 8.6, “The J2EE component” on page 278

Table 8-3 provides the details of the Pet Store environment needed to configure and deploy the needed TMTP components, and Figure 8-59 shows the details of defining/deploying the Management Agent on a Weblogic 7.0 application server.

Table 8-3 Pet Store J2EE configuration parameters

Field	Default value
Application Server Name	petstoreServer
Application Server Home	c:\bea\weblogic700
Domain	petstore
Java Home	c:\bea\jdk131_03
Start with Script	check
Domain Path	c:\bea\weblogic700\samples\server\config\petstore\
Path and file name	c:\bea\weblogic700\samples\server\config\petstore\startPetStore.cmd

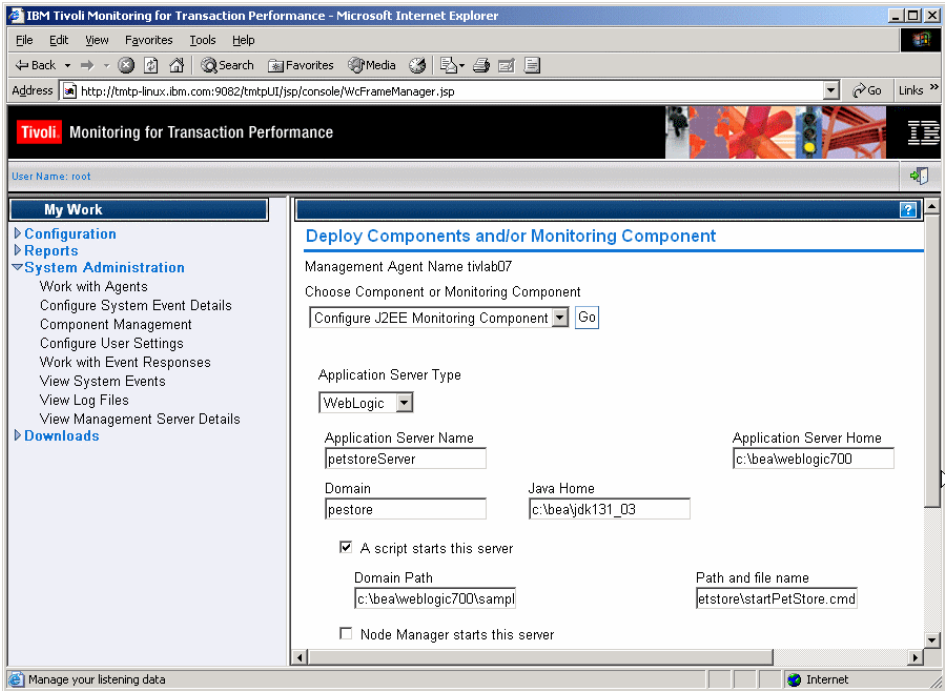


Figure 8-59 Weblogic Management Agent configuration

8.8.3 J2EE discovery and listening policies for Weblogic Pet Store

After successful installation of the Management Agent onto the Weblogic application server, the next steps are creating the agent groups, schedules, and discovery and listening policies.

For details on how to create discovery and listening policies, please refer to 8.6.2, “J2EE component configuration” on page 282.

1. We have created discovery policy petstore_j2ee_dis with the following configuration capturing data from the Pet Store application that generated by all users:

   URI Filter http://.*/petstore/.*

   User name .*

   In addition, a schedule for discovery and listening policies has been created. The name of the schedule is petsore_j2ee_dis_forever, and it runs continuously.

   Note: Before creating the listening policies for the J2EE applications, it is important to create a discovery policy and browse the Pet Store application and generate some transactions.

2. The J2EE listening policy named petstore_j2ee_lis has been defined to listen for Pet Store transactions to the URI `http://tivlab01.itsc.austin.ibm.com:7001/petstore/product.screen?category_id=FISH`, as shown in Figure 8-60 on page 313.

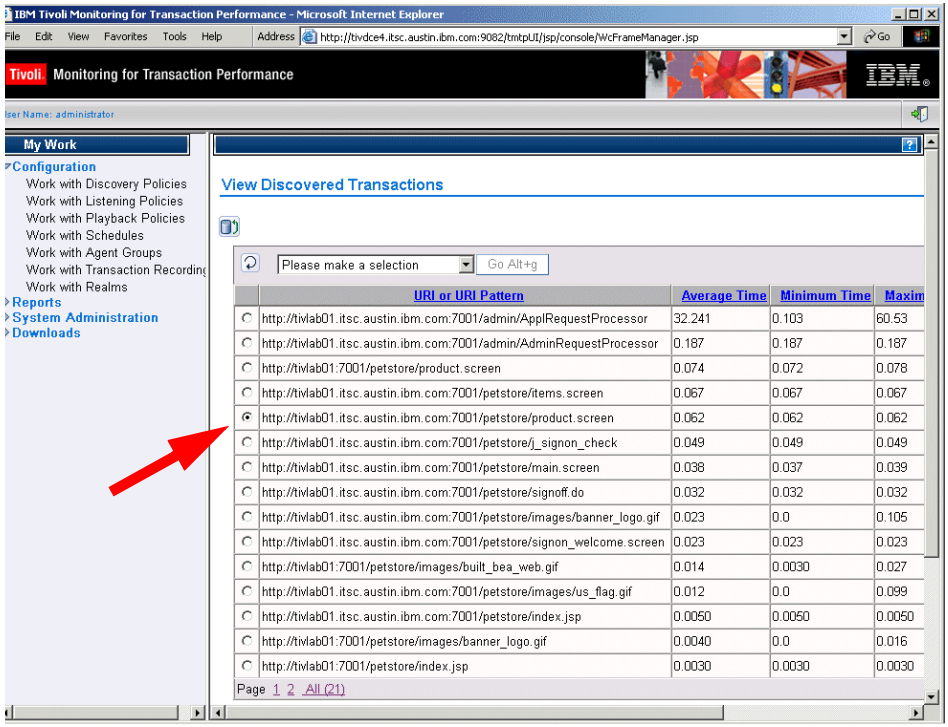


Figure 8-60 Creating listening policy for Pet Store J2EE Application

The average response time reported by the discovery policy is 0.062 seconds (see Figure 8-61 on page 314).

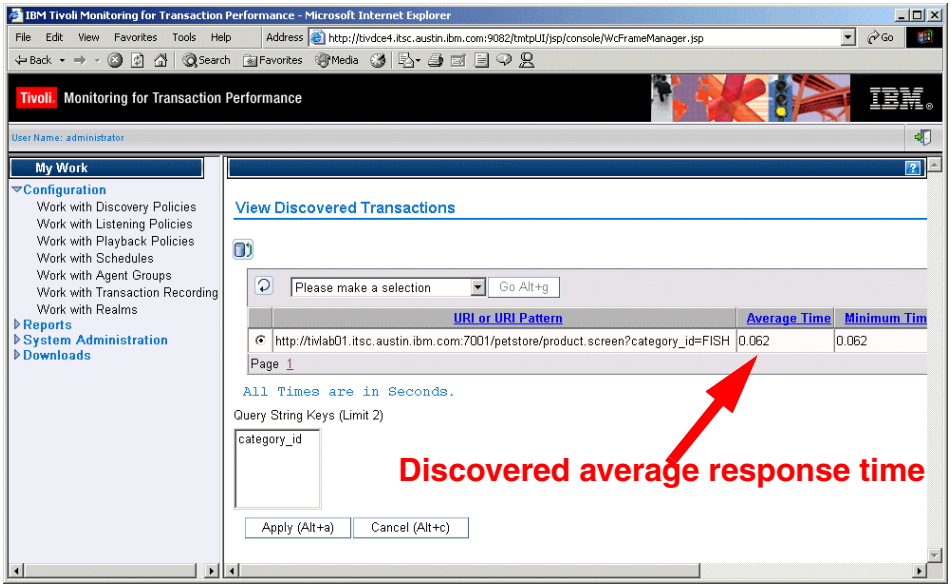


Figure 8-61 Choose Pet Store transaction for Listening policy

A threshold is defined for the listening policy for response times 20% higher than the average reported by the discovery policy, as shown in Figure 8-62.

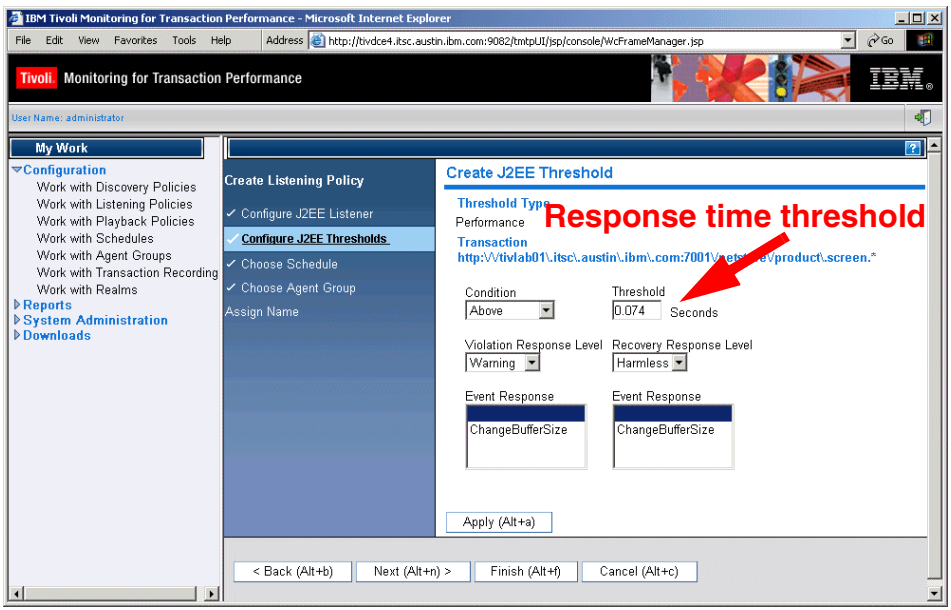


Figure 8-62 Automatic threshold setting for Pet Store

Quality of Service listening policy for Pet Store

To define a QoS listening policy for the Pet Store application (pestore_qos_lis), we used the following transaction filter:

```
http:\/\/tivlab01\.itsc\.austin\.ibm\.com:80\/petstore\/signon_welcome\.screen.
*
```

Settings for the Back-End Service Time threshold are shown in Figure 8-63.

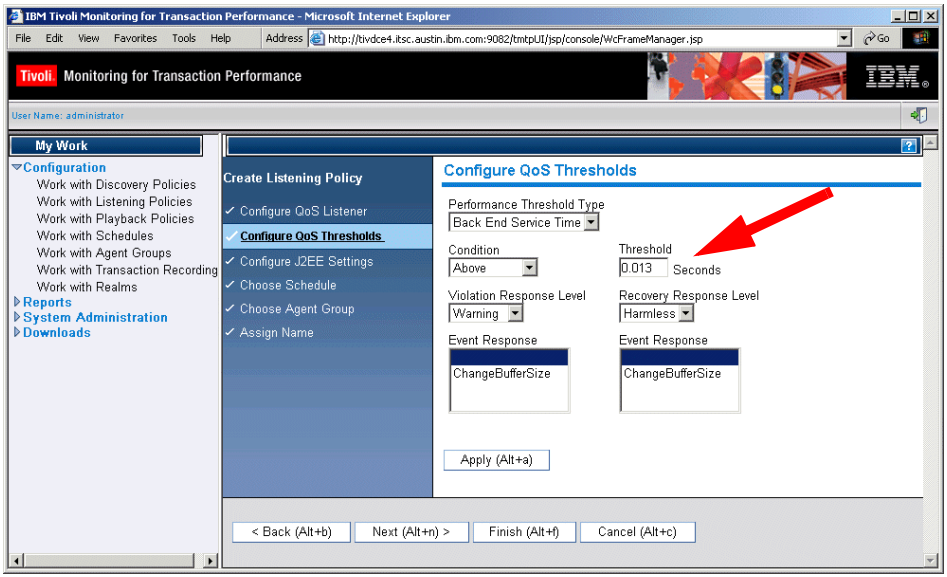


Figure 8-63 QoS listening policies for Pet Store automatic threshold setting

In addition, we provided the J2EE settings for the QoS listening policy shown in Figure 8-64 on page 316 in order to ensure correlation between the QoS front-end monitoring and the back-end monitoring provided by the J2EE component.

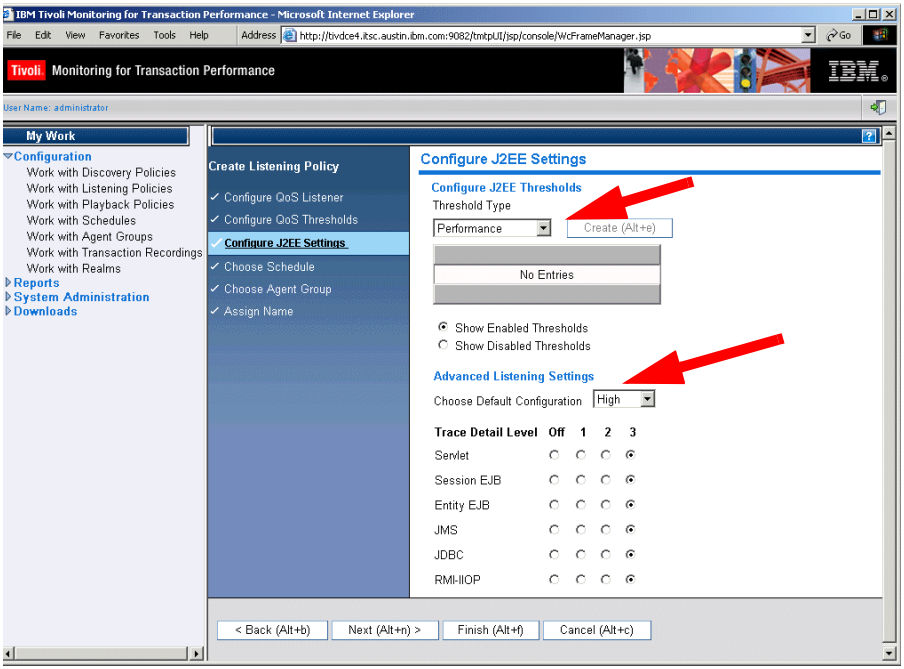


Figure 8-64 QoS correlation with J2EE application

8.8.4 Event analysis and online reports for Pet Store

If we analyze the Pet Store business process from login to submit from the Pet Store Web site, we have a total of nine steps:

1. Log in to Pet Store site
2. Select pet
3. Select product category
4. Select/view items for this product category
5. Add to cart
6. View the shopping cart
7. Proceed to checkout
8. Supply order information
9. Submit

STI, QoS, and J2EE combined scenario

We want to find the User Experienced Time and the Back-End Service Time for end-users buying pets the e-business way. Since we cannot control the behavior of users, STI is used to run the same transaction consistently.

To facilitate this, an STI playback policy is created to run a simulated Pet Store transaction named petstore_2_order. Petstore_2_order is configured to allow correlation with the back-end J2EE monitoring.

The Transaction with Subtransaction report shown in Figure 8-65 shows that the total simulated end-user response time for Pet Store playback policy is 8.12 sec. It also shows that five subtransactions has been executed, and that subtransaction number 3 is responsible for the biggest part of the total response time. This report is very helpful to in order to identify, over a longer period of time, which subtransaction traditionally contributes most to the overall response time.

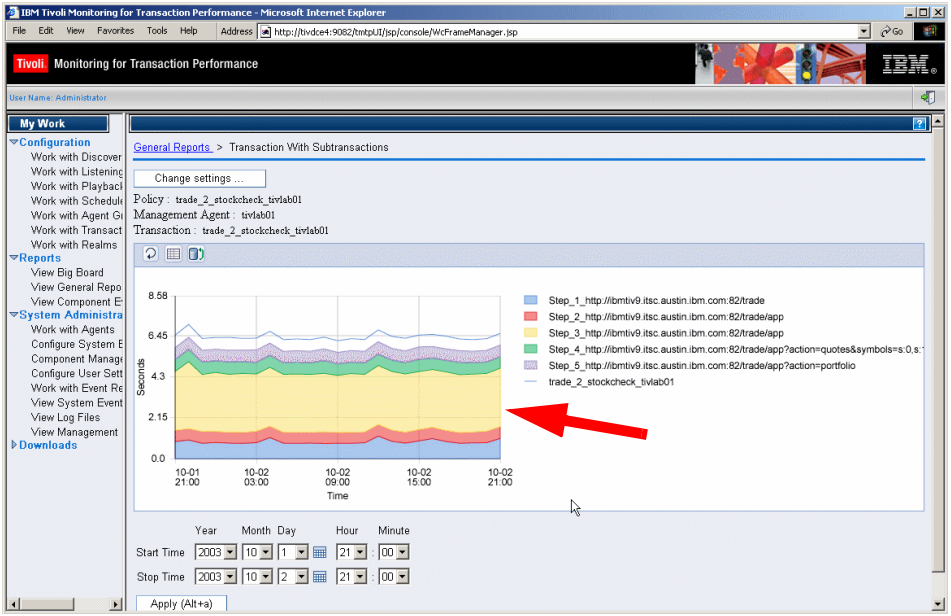


Figure 8-65 Pet Store transaction and subtransaction response time by STI

From the Page Analyzer Viewer report shown in Figure 8-66 on page 318, we can see that the enter_order_information_screen subtransaction takes longer (2.4 seconds) to present the output to the end user. By using Page Analyzer Viewer, we can find out (for STI transactions) which subtransactions take a long time and what type of function is involved. Among the functions that can be identified are:

- DNS resolution

- Connection
- Connection idle
- Socket connection
- SSL connection
- Server response error

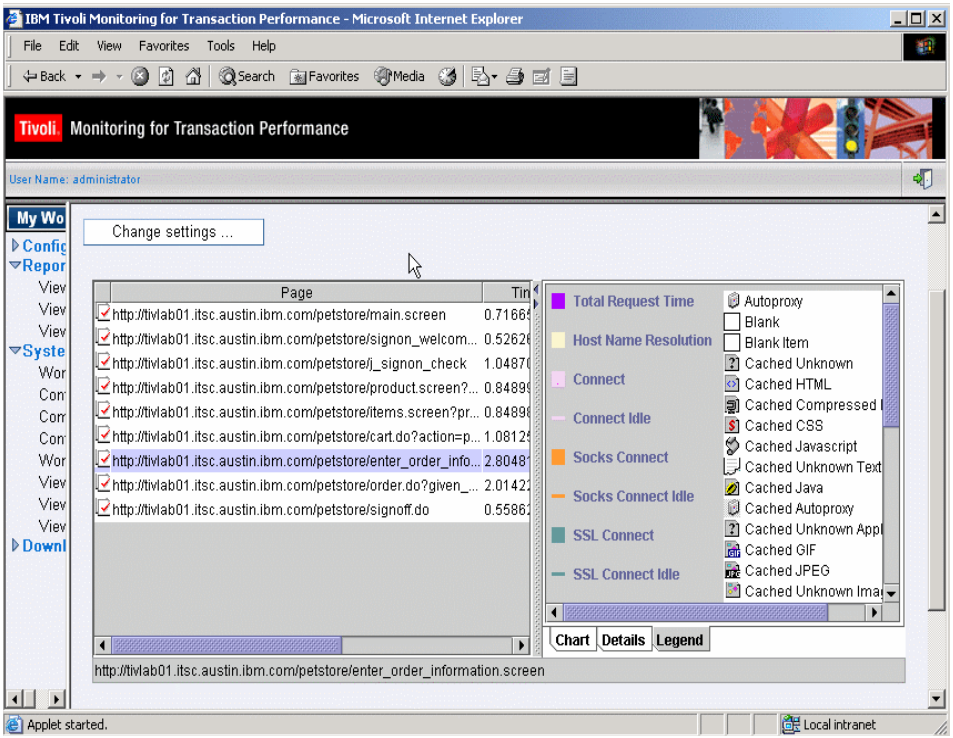


Figure 8-66 Page Analyzer Viewer report of Pet Store business transaction

The topology view in Figure 8-67 on page 319 shows how the STI transactions propagates to the J2EE Application Server and shows the parent/child relationship with the Pet Store simulated transaction and various J2EE application components.

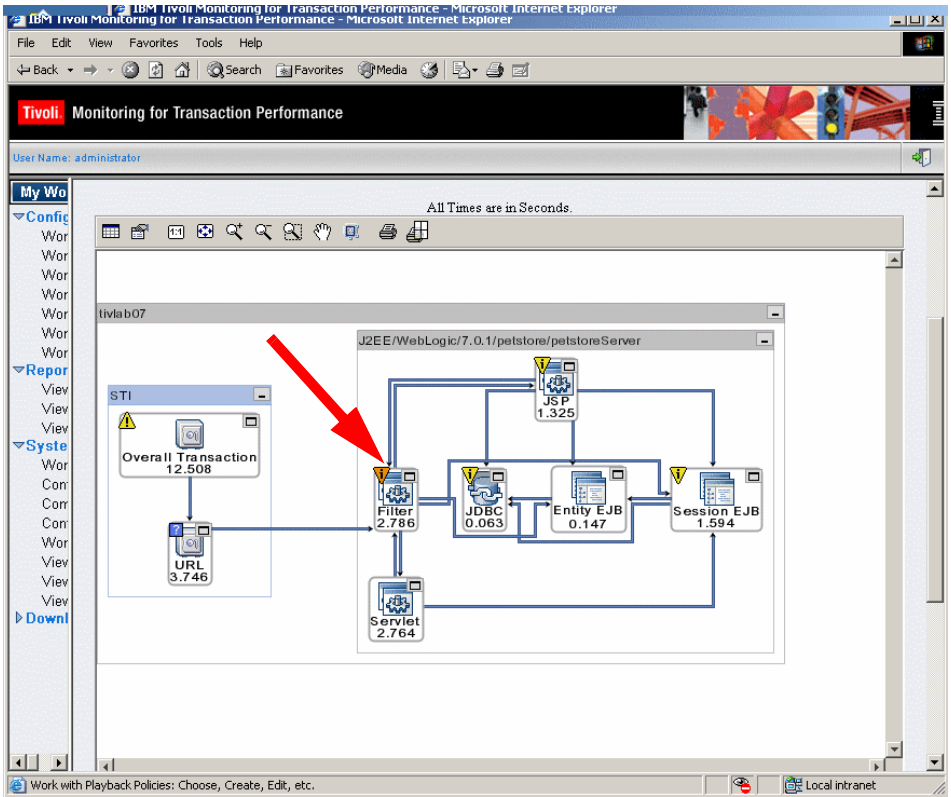


Figure 8-67 Correlation of STI and J2EE view for Pet Store application

With respect to the thresholds defined for the QoS and J2EE listening policies in this scenario, we see from Figure 8-68 on page 320 (the aggregate topology view) that threshold violations have been identified and reported (Most_violated) in the report.

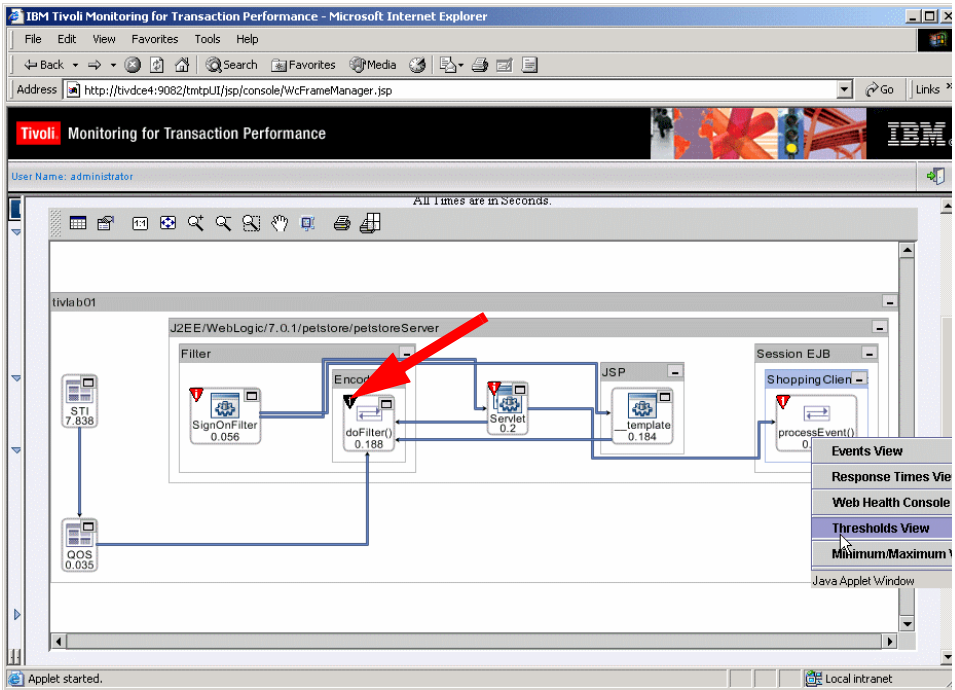


Figure 8-68 J2EE dofilter() methods creates events

Pet Store J2EE performance scenario

We want to identify the performance characteristics of different J2EE application components (such as Pet Store JSP, Servlets, EJB, and JDBC) during business hours, especially during peak hours. In addition we want to identify the application's bottleneck and the component responsible in order to figure out if the application is under- or over-provisioned. Furthermore, we want to find the real Back-End Service Time for all back-end components and the Round Trip Time for an end-user.

A J2EE listening policy is created and named petstore_j2ee_lis to capture specific Pet Store business transactions.

A QoS listening policy is created and named petstore_qos_lis to capture the real response time with the J2EE application components response for specific transactions against the Pet Store site.

Please refer to 7.1, "Reporting overview" on page 212 for details on how to use the online reports in IBM Tivoli Monitoring for Transaction Performance Version 5.2.

From the J2EE topology view shown in Figure 8-69, we see that SessionEJB indicates an alert. If we drill down in the SessionEJB, we realize that the getShoppingClienFacade method is responsible for this violation, as shown in see Figure 8-70 on page 322.

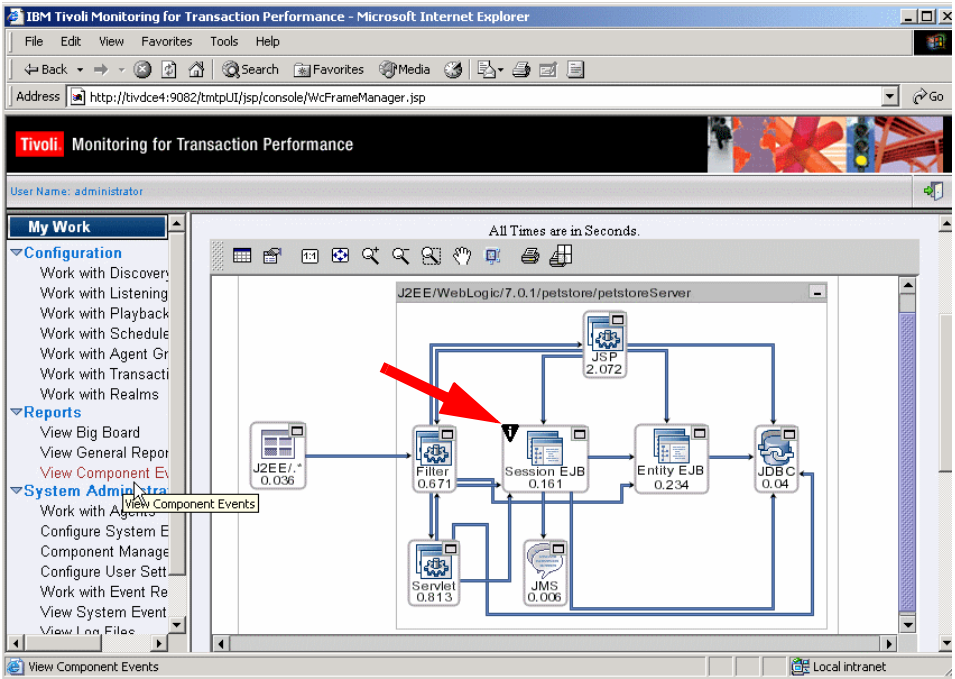


Figure 8-69 Problem indication in topology view of Pet Store J2EE application

From the topology view, we can jump directly to the Response Time View for the particular application component, as shown in Figure 8-70 on page 322, in order to get the report shown in Figure 8-71 on page 322.

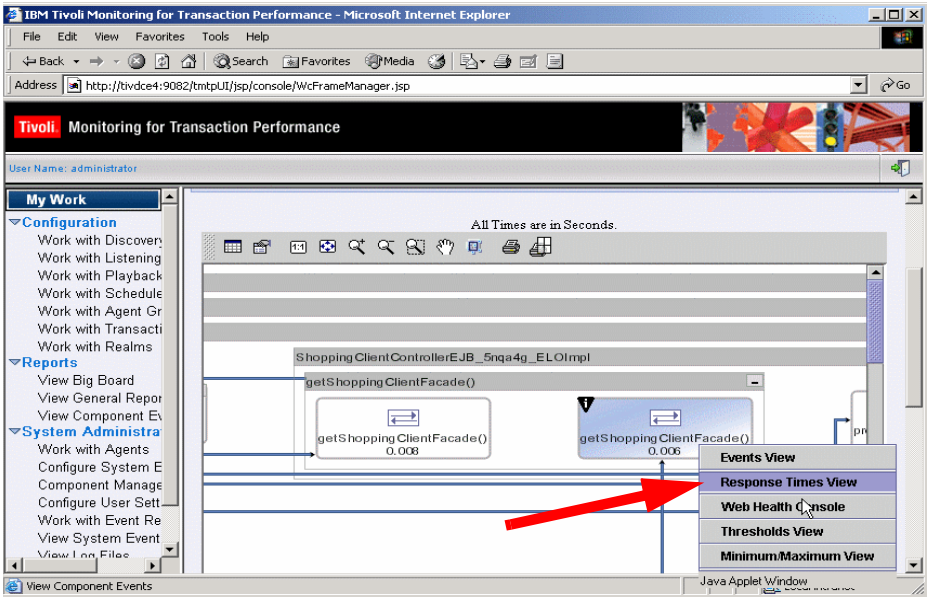


Figure 8-70 Topology view: event violation by getShoppingClientFacade

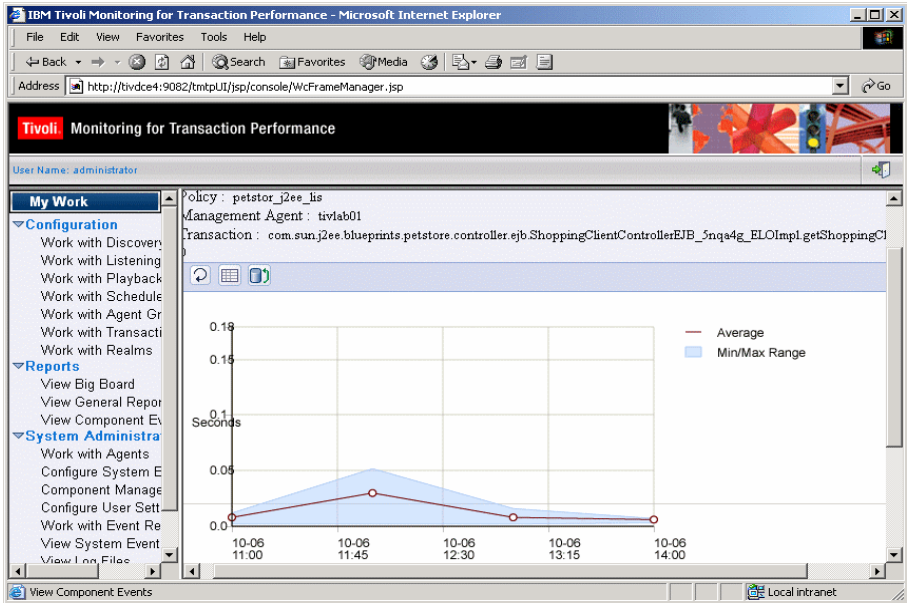


Figure 8-71 Response time for getShoppingClienFacade method

Finally, the real-time transaction performance (total Round Trip Time and Back End Service Time) of the Pet Store site, as well as J2EE components response time, are shown in Figure 8-72.

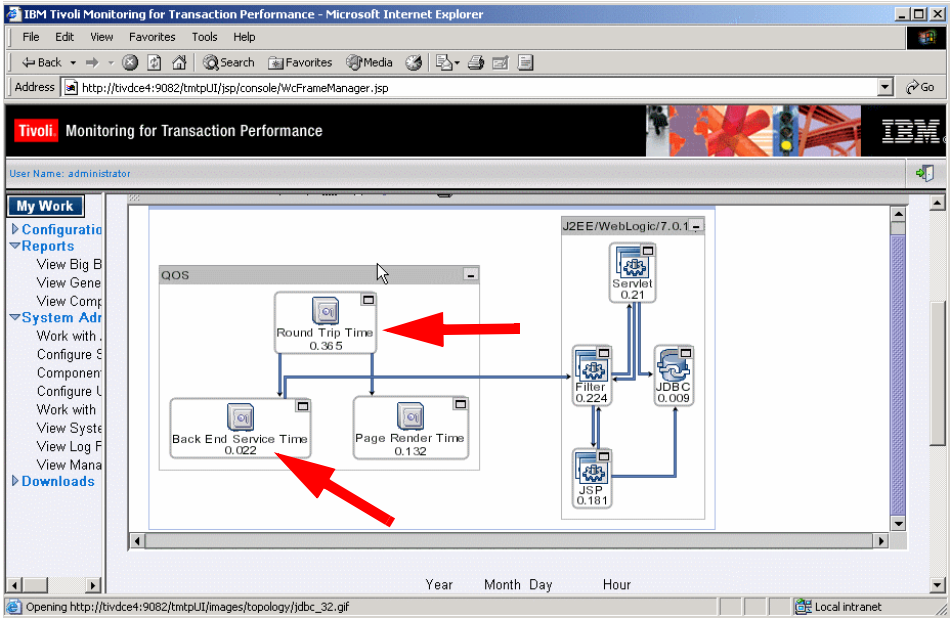


Figure 8-72 Real-time Round Trip Time and Back-End Service Time by QoS

9

Rational Robot and GenWin

This chapter demonstrates how to use the Rational Robot to record e-business transactions, how to instrument those transactions in order to generate relevant e-business transaction performance data, and how to use TMTP's GenWin facility to manage playback of your transactions.

© Copyright IBM Corp. 2003. All rights reserved.

9.1 Introducing Rational Robot

Rational Robot is a collection of applications that can be used to perform a set of operations on a graphical interface or to operate directly at the network protocol layer using an intuitive and easy to use interface.

Rational Robot has been around a while and is reliable and complete in the features it offers, the range of supported application types is considerable, and the behavior between application types is almost identical.

It provides a robust programming interface that allows you to add strict controls to the program flow and includes technologies that allows the simulation to complete, even if portions of the graphical interface of the application under stress changes during development.

Each record step is shown graphically with a specific iconography.

Rational Robot can be used to simulate transactions on applications running in generic Windows environment, Visual Basic applications, Oracle Forms, Powerbuilder applications, Java applications, Java applets, or Web sites. Some of these applications are supported out of the box, others require the installation of specific Application Enablers provided with Rational Robot, and still others require the user to load a specific Application Extension.

It allows for quick visual recording of the application under test and playback in a debugging environment to ensure that the simulation flows correctly.

Scripts can be played back on a variety of Windows platforms, including Windows NT® 4.0, Windows XP, Windows 2000, Windows 98, and Windows Me.

9.1.1 Installing and configuring the Rational Robot

Rational Robot is provided by TMTP Version 5.2 as a zip file that containing the Rational Robot CD iso image so that you can burn your own Rational Robot CD using your favorite software. The setup procedure does not differ if the image is used from the CD or downloaded from TMTP.

Rational Robot is installed following the generic setup steps you need to follow on most Windows applications. After the installation there are specific steps you must follow to enable and load all the components needed to record and playback a simulation on the application you will use (Java, HTML, and so on).

Installing

Put the Rational Robot CD-ROM in the CD-ROM tray of the machine where simulations will be recorded or played back; setup is identical in both cases.

Double click on the C517JNA.exe application, which you can find in the robot2003GA folder in the Rational Robot CD. The setup procedure will start. You should get the window shown in Figure 9-1.

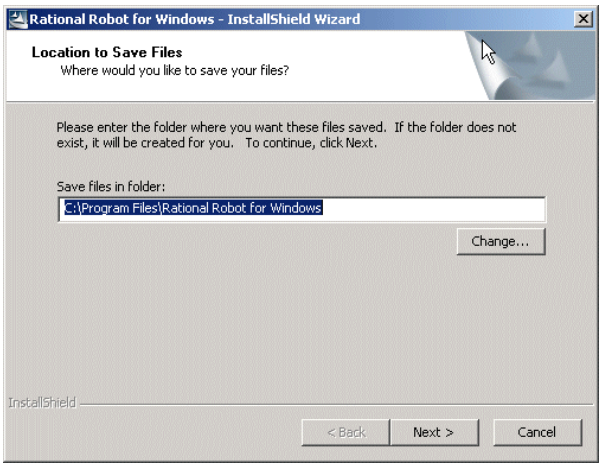

Rational Robot for Windows - InstallShield Wizard
Location to Save Files
Where would you like to save your files?
Please enter the folder where you want these files saved. If the folder does not exist, it will be created for you. To continue, click Next.
Save files in folder:
C:\Program Files\Rational Robot for Windows
Change...
InstallShield
< Back
Next >
Cancel

Figure 9-1 Rational Robot Install Directory

Change the install directory if you are not satisfied with the default setting and select **OK**. The install directory will be displayed at a later stage, but no changes will be possible. After you click **Next**, the install continues for a while (see Figure 9-2 on page 328).

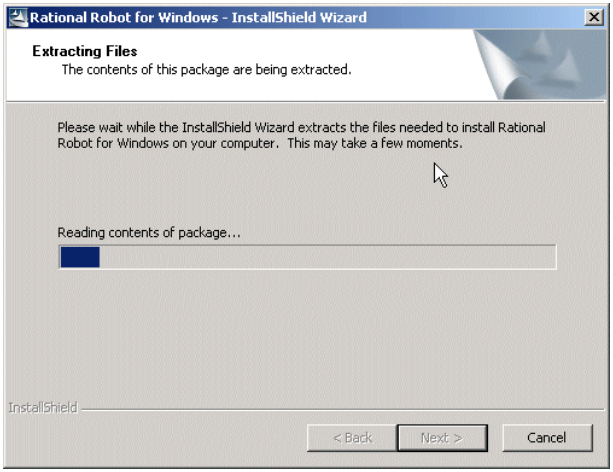

Rational Robot for Windows - InstallShield Wizard
Extracting Files
The contents of this package are being extracted.
Please wait while the InstallShield Wizard extracts the files needed to install Rational Robot for Windows on your computer. This may take a few moments.
Reading contents of package...
InstallShield
< Back
Next >
Cancel

Figure 9-2 Rational Robot installation progress

The setup wizard will be loaded and displayed (see Figure 9-3).

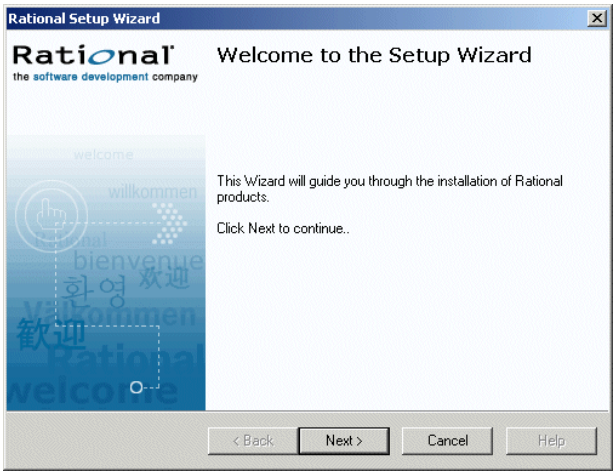


Figure 9-3 Rational Robot Setup wizard

Click on **Next**, and the Product Selection panel is displayed. In this panel, you have the choice of selecting the Rational License Manager that you need to use Robot and Rational Robot itself. Select **Rational Robot** in the left pane (see Figure 9-4 on page 329).

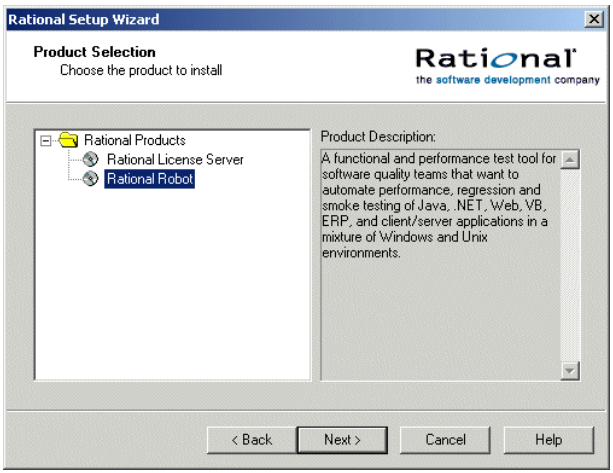

Rational Setup Wizard
Product Selection
Choose the product to install
Rational
the software development company
Rational Products
Rational License Server
Rational Robot
Product Description:
A functional and performance test tool for software quality teams that want to automate performance, regression and smoke testing of Java, .NET, Web, VB, ERP, and client/server applications in a mixture of Windows and Unix environments.
< Back
Next >
Cancel
Help

Figure 9-4 Select Rational Robot component

Click **Next** to continue the setup; the Deployment Method panel is displayed (see Figure 9-5).

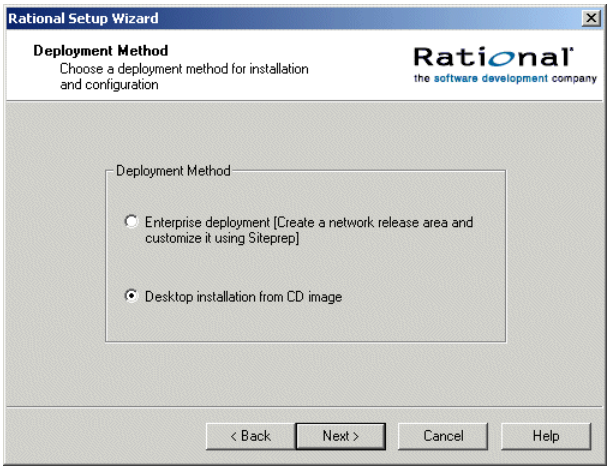


Figure 9-5 Rational Robot deployment method

Select **Desktop installation from CD image** and click on **Next**; the installation will check various items and then display the Rational Robot Setup Wizard (see Figure 9-6 on page 330).

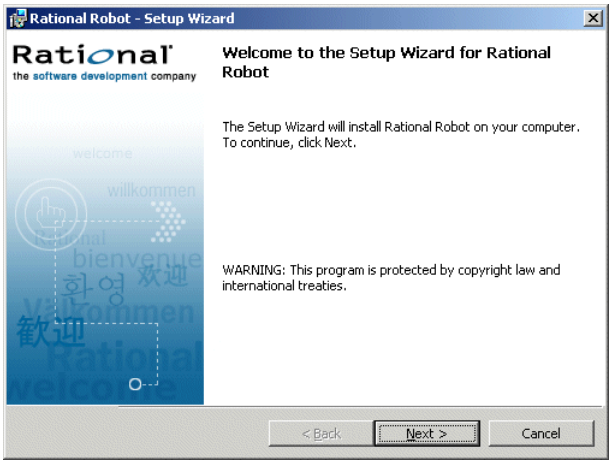

Rational Robot - Setup Wizard
Rational
the software development company
Welcome to the Setup Wizard for Rational Robot
The Setup Wizard will install Rational Robot on your computer. To continue, click Next.
WARNING: This program is protected by copyright law and international treaties.
< Back
Next >
Cancel

Figure 9-6 Rational Robot Setup Wizard

Click on **Next**; the Product Warnings will be displayed (see Figure 9-7).

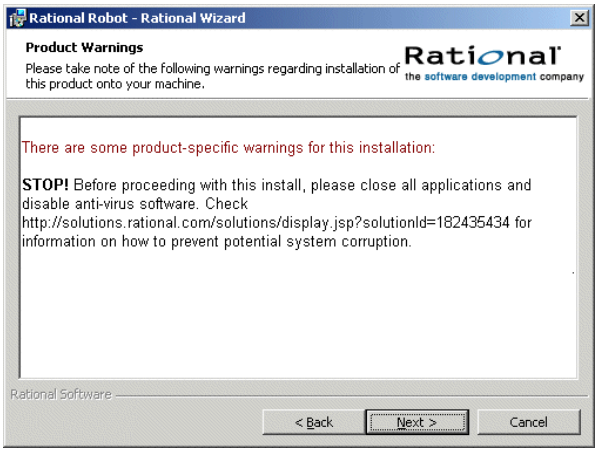

Rational Robot - Rational Wizard
Product Warnings
Please take note of the following warnings regarding installation of this product onto your machine.
Rational
the software development company
There are some product-specific warnings for this installation:
STOP! Before proceeding with this install, please close all applications and disable anti-virus software. Check http://solutions.rational.com/solutions/display.jsp?solutionId=182435434 for information on how to prevent potential system corruption.
Rational Software
< Back
Next >
Cancel

Figure 9-7 Rational Robot product warnings

Check if any message is relevant to you. If you already have Rational products installed, you could be required to upgrade those products to the latest version.

Click on **Next**; the License Agreement panel will be displayed (see Figure 9-8 on page 331).

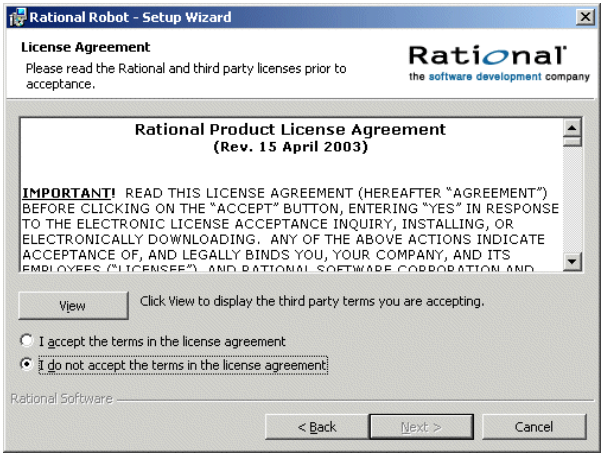

Rational Robot - Setup Wizard
License Agreement
Please read the Rational and third party licenses prior to acceptance.
Rational
the software development company
Rational Product License Agreement
(Rev. 15 April 2003)
IMPORTANT! READ THIS LICENSE AGREEMENT (HEREAFTER "AGREEMENT") BEFORE CLICKING ON THE "ACCEPT" BUTTON, ENTERING "YES" IN RESPONSE TO THE ELECTRONIC LICENSE ACCEPTANCE INQUIRY, INSTALLING, OR ELECTRONICALLY DOWNLOADING. ANY OF THE ABOVE ACTIONS INDICATE ACCEPTANCE OF, AND LEGALLY BINDS YOU, YOUR COMPANY, AND ITS
View
Click View to display the third party terms you are accepting.
I accept the terms in the license agreement
I do not accept the terms in the license agreement
Rational Software
< Back
Next >
Cancel

Figure 9-8 Rational Robot License Agreement

Select **I accept the terms in the license agreement** radio button, and then click on **Next**; the Destination Folder panel is displayed (see Figure 9-9).

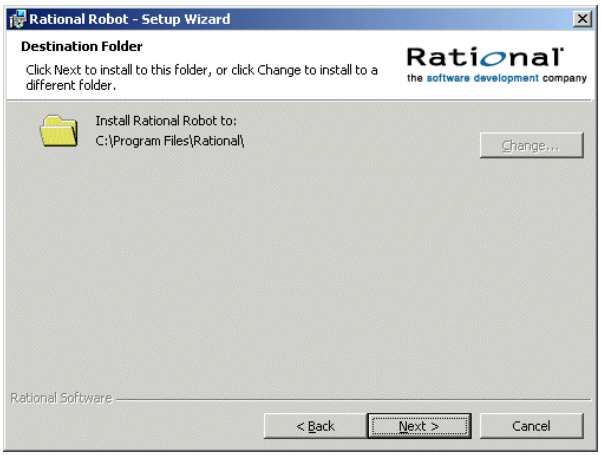


Figure 9-9 Destination folder for Rational Robot

Click on **Next**; the install folder cannot be changed at this stage. The Custom Setup panel is displayed. Leave the defaults and click on **Next**; the Ready to Install panel is displayed (see Figure 9-10 on page 332).

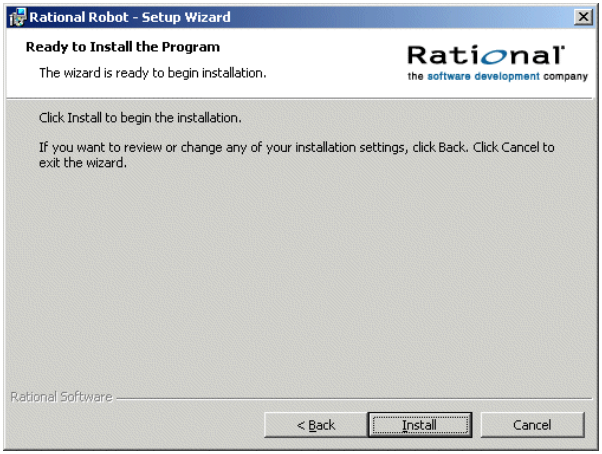

Rational Robot - Setup Wizard
Ready to Install the Program
The wizard is ready to begin installation.
Rational
the software development company
Click Install to begin the installation.
If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.
Rational Software
< Back
Install
Cancel

Figure 9-10 Ready to install Rational Robot

You can now click on **Next** to complete the setup. After a while, the Setup Complete dialog is displayed (see Figure 9-11).

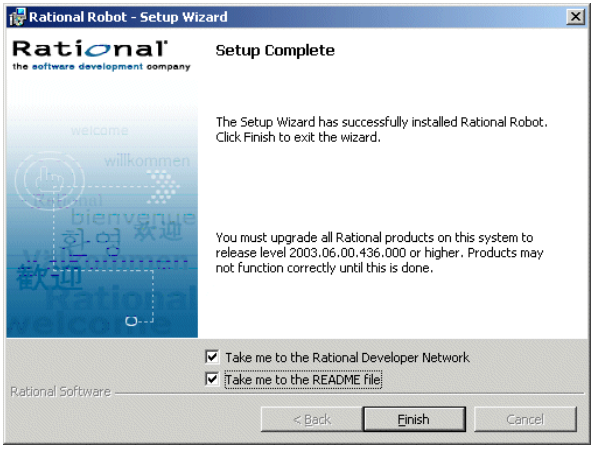

Rational Robot - Setup Wizard
Rational
the software development company
Setup Complete
The Setup Wizard has successfully installed Rational Robot. Click Finish to exit the wizard.
You must upgrade all Rational products on this system to release level 2003.06.00.436.000 or higher. Products may not function correctly until this is done.
Take me to the Rational Developer Network
Take me to the README file
Rational Software
< Back
Finish
Cancel

Figure 9-11 Rational Robot setup complete

Deselect the check boxes if you want and click on **Finish**.

Installing the Rational Robot hotfix

There is a hotfix provided in the Rational Robot CD under the folder robot2003Hotfix. You can install it by doing the following:

1. Close Rational Robot if you are already running it.

2. Search the folder where Rational Robot has been installed for the file rtrobo.exe. Copy the rtrobo.exe file and the CLI.bat files provided in the robot2003Hotfix folder into the folder where you found rtrobo.exe.
3. Open a command prompt in the Rational Robot install folder and run CLI.bat. This is just a test script; if you do not get any errors, the fix is working OK and you can close the command prompt.

Installing the Rational License Server

Repeat all the steps in the above section, but select the Rational License Server in the Product Selection panel. Complete the installation as you did with Rational Robot.

After setting up the Rational License Server, you can install the named-user license provided in the Rational Robot CD.

Installing the Rational Robot License

4. To install the named-user license you have to start the Rational License Key Administrator by selecting **Start** → **Programs** → **Rational Software** and clicking on the **License Key Administrator** icon.

The License Key Administrator starts and displays a wizard (see Figure 9-12).

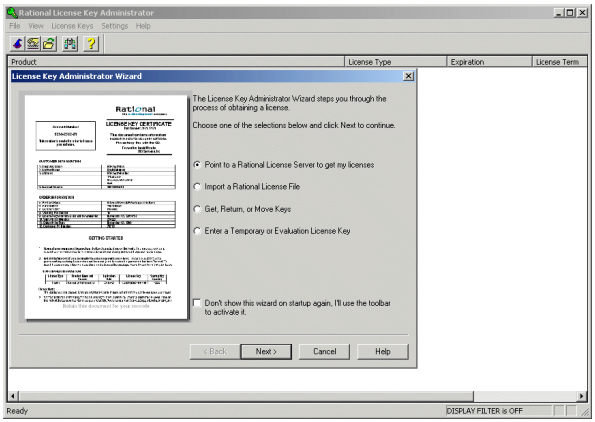

Rational License Key Administrator
File View License Keys Settings Help
Product
License Type
Expiration
License Term
License Key Administrator Wizard
The License Key Administrator Wizard steps you through the process of obtaining a license.
Choose one of the selections below and click Next to continue.
Point to a Rational License Server to get my licenses
Import a Rational License File
Get, Return, or Move Keys
Enter a Temporary or Evaluation License Key
Don't show this wizard on startup again, I'll use the toolbar to activate it.
< Back
Next >
Cancel
Help
Ready
DISPLAY FILTER is OFF

Figure 9-12 Rational Robot license key administrator wizard

In the License Key Administrator Wizard, select **Import a Rational License File** and click on **Next**. The Import License File panel is displayed; click the **Browse** button and select the ibm_robot.upd provided in the root folder of the Rational Robot CD (see Figure 9-13 on page 334).

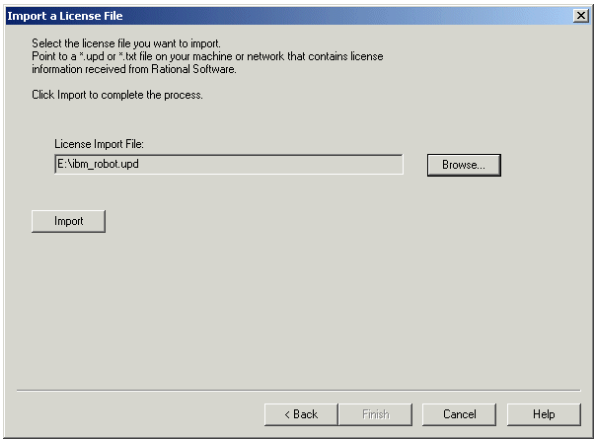

Import a License File
Select the license file you want to import.
Point to a *.upd or *.txt file on your machine or network that contains license information received from Rational Software.
Click Import to complete the process.
License Import File:
E:\ibm_robot.upd
Browse...
Import
< Back
Finish
Cancel
Help

Figure 9-13 Import Rational Robot license

Click on the **Import** button to import the license. The Confirm Import panel is displayed (see Figure 9-14).

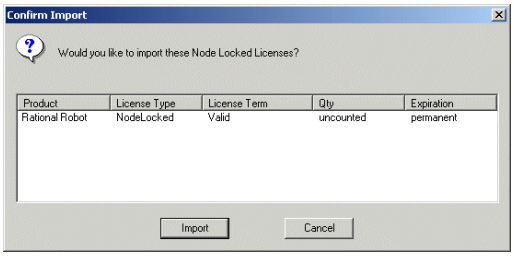


Figure 9-14 Import Rational Robot license (cont...)

Click on the **Import** button on the Confirm Import panel to import the IBM license in the License Key Manager; if the import process is successful, you will se a confirmation message box (see Figure 9-15).

Figure 9-15 Rational Robot license imported successfully

Click on **OK** to return to the License Key Manager.

The License Key Manager will now display the new license as being available (see Figure 9-16).

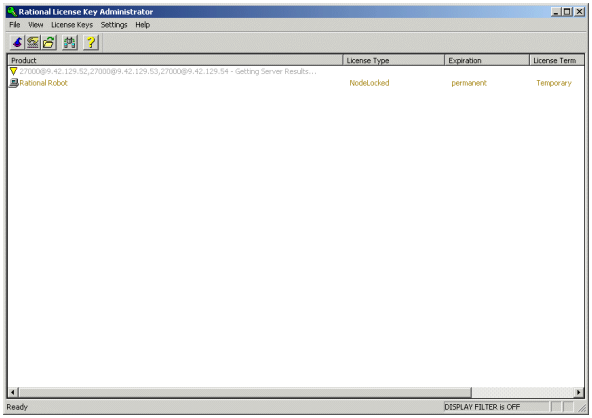


Figure 9-16 Rational Robot license key now usable

You can now close the License Key Administrator. Rational Robot is now ready for use.

Configuring Rational Robot Java Enabler and Extensions

For Rational Robot to correctly simulate operations being performed on Java applications, the Java Extension must be loaded and a specific component called Robot Java Enabler must be installed and configured.

Configuring the Java Enabler

The Java Enabler setup program is installed during the Rational Robot installation, but has to be selected and customized for use before you can record a simulation successfully. It is important that you ensure that Rational Robot is *not* running when you set up the Java Enabler; you will need to enable any JVM you add to the system and need to use.

You can set up the Java Enabler by selecting the **Java Enabler** setup icon, which you can find by selecting **Start** → **Rational Software** → **Rational Test** program group.

After selecting the Java Enabler icon, the setup starts and a dialog with a selection of Java Enabler Types is displayed (see Figure 9-17 on page 336).

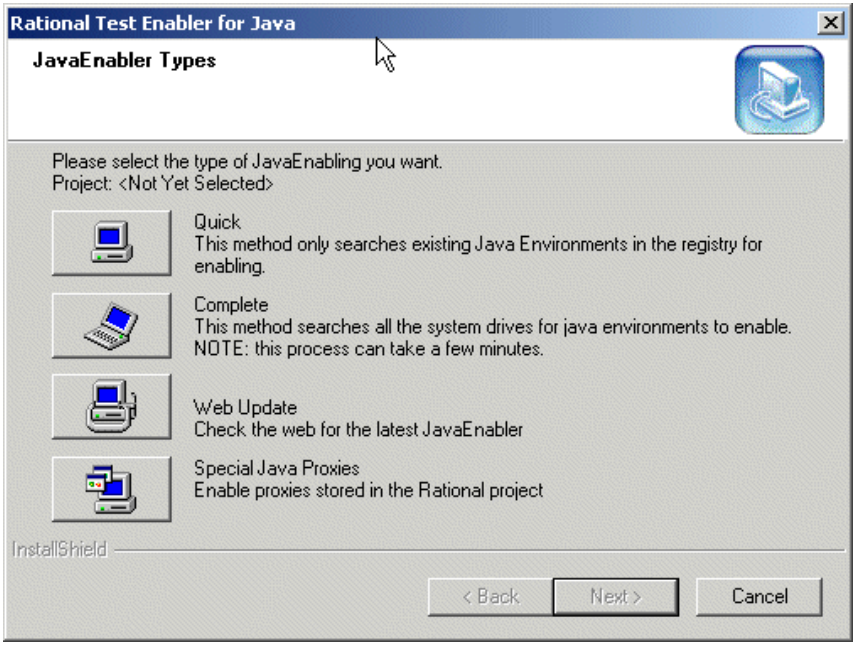

Rational Test Enabler for Java
JavaEnabler Types
Please select the type of JavaEnabling you want.
Project: <Not Yet Selected>
Quick
This method only searches existing Java Environments in the registry for enabling.
Complete
This method searches all the system drives for java environments to enable.
NOTE: this process can take a few minutes.
Web Update
Check the web for the latest JavaEnabler
Special Java Proxies
Enable proxies stored in the Rational project
InstallShield
< Back
Next >
Cancel

Figure 9-17 Configuring the Rational Robot Java Enabler

Select the **Quick** setup method to enable Rational Robot for the JVM in use. If you have multiple JVMs and want to be sure that you enable all of them for Rational Robot, you can instead select **Complete**, and this will perform a full scan of your hard drive for all installed JVMs.

After selecting **Quick**, a dialog will be displayed with the JVMs found on the system (see Figure 9-18 on page 337). From this list, you should select the JVM you will use with the simulations and select **Next**.

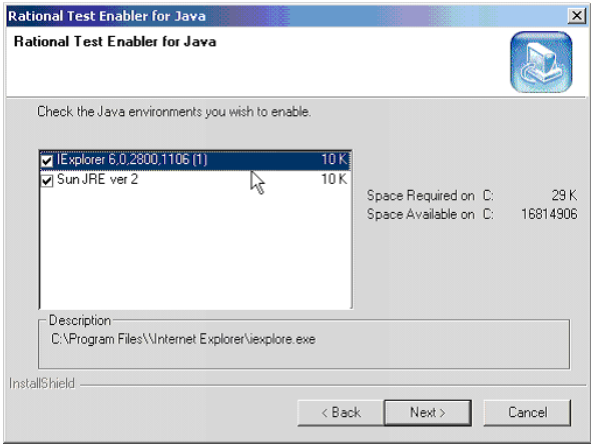

Rational Test Enabler for Java
Rational Test Enabler for Java
Check the Java environments you wish to enable.
IExplorer 6,0,2800,1106 (1) 10 K
Sun JRE ver 2 10 K
Space Required on C: 29 K
Space Available on C: 16814906
Description
C:\Program Files\\Internet Explorer\iexplore.exe
InstallShield
< Back
Next >
Cancel

Figure 9-18 Select appropriate JVM

The setup completes and you are given an option to verify the setup log. The log will show what files have been changed/copied during the setup process.

Rational Robot is now ready to record and playback simulations on Java applications running in the JVM that you enabled.

If you add a new JVM or change the JVM you initially enabled, you will have to re-run the Rational Test Enabler on the new JVM.

Loading the Java Extension

The Java enabler, although important, is not the only component needed to record simulations on Java applications: a specific enabler has to be loaded when Rational Robot starts.

The Java Enabler is loaded by default after Rational Robot is installed; to ensure that it is being loaded, select **Tools** → **Extension Manager** in the Rational Robot menu. The Extension Manager dialog is displayed (see Figure 9-19 on page 338).

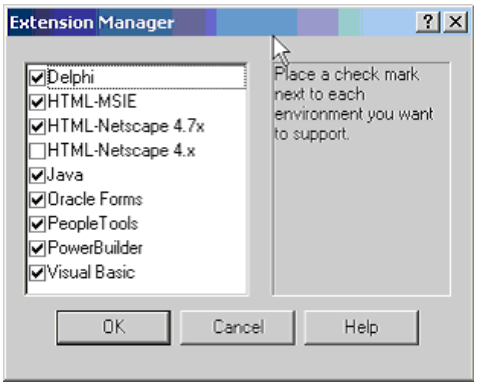


Figure 9-19 Select extensions

Ensure that the Java check box is selected; if it was not, you would also need to restart Rational Robot to load the Java Extension.

Loaded Application Extensions loaded have a performance downgrade drawback: if you are not writing simulations on the other application types in the list, deselect them.

Setting up the HTML extensions

Rational Robot supports simulations that run in a Web browser, thanks to browser specific extensions that must be loaded by Rational Robot.

The browsers supported for testing are all versions of Microsoft Internet Explorer, Netscape 4.x and Netscape 4.7x.

By default, Rational Robot supports MSIE and Netscape 4.7x. You can check the loaded extensions by selecting **Tools** → **Extension Manager**; this will display the Extension Manager dialog shown in Figure 9-19.

Any changes in the Extension Manager list will require Rational Robot to restart in order to load the selected extensions.

If you plan to test only a specific set of the application types listed in the Extension Manager, deselect those you do not plan to use to increase Rational Robots performance.

One important point to consider when planning a simulation in a browser is that the machine that will run the simulation's browser must be of the same kind and use the same settings as the one where the simulation is recorded: A typical error is to have different settings for the cookies so that one browser accepts

them while the other displays a dialog to the user, thus breaking the simulation flow.

Differences for Netscape users

We recommend using Netscape 4.x only if it is specifically needed, since it requires local browser caching to be enabled and would not simulate applications using HTTPS. Also, Netscape 4.7x and Netscape 4.x are mutually exclusive; if you want to use one, you should *not* select the other.

9.1.2 Configuring a Rational Project

Before you can record a Rational Script, you must have a valid Rational Project to use. During Rational Robot installation, you will be taken through the following procedure. However, you will also have to use this procedure to create a Rational Robot project for use by the Generic Windows Management Agent.

First, you need to decide on the location of your project . All Rational Projects are stored in specific directory structures, and the top-level directory for each project has to be created manually before defining the project to Rational. When using Rational with the TMTP Generic WIndows Management Agent, the project directory has to be available to the Generic Windows Management Agent. The base location for all projects are dictated by the Generic Windows Management Agent to be the $MA\apps\genwin\ directory (where $MA denotes the installation directory of the Management Agent). Since this directory structure is created as part of the Generic Windows Management Agent installation procedure, we advise you to install this component prior to defining and recording projects.

Before proceeding, either install the Generic Windows Management Agent, or open Windows Explorer and create the directory structure for the project. Make sure the project directory itself is empty.

To create a Rational Project, perform the following steps:

1. Start the Rational Administrator by selecting **Start** → **Programs** → **Rational Robot** → **Rational Administrator**.
2. Start the New Project Wizard by clicking **File** → **New Project** on the Administrator menu.
3. On the wizard's first page (Figure 9-20 on page 340):
   a. Supply a name for your project, for example, testscripts. The dialog box prevents you from typing illegal characters.
   b. In the Project Location field, specify a UNC path to the root of the project, referring to the directory name you created in above. It does not really have to be a shared network directory with a UNC path.

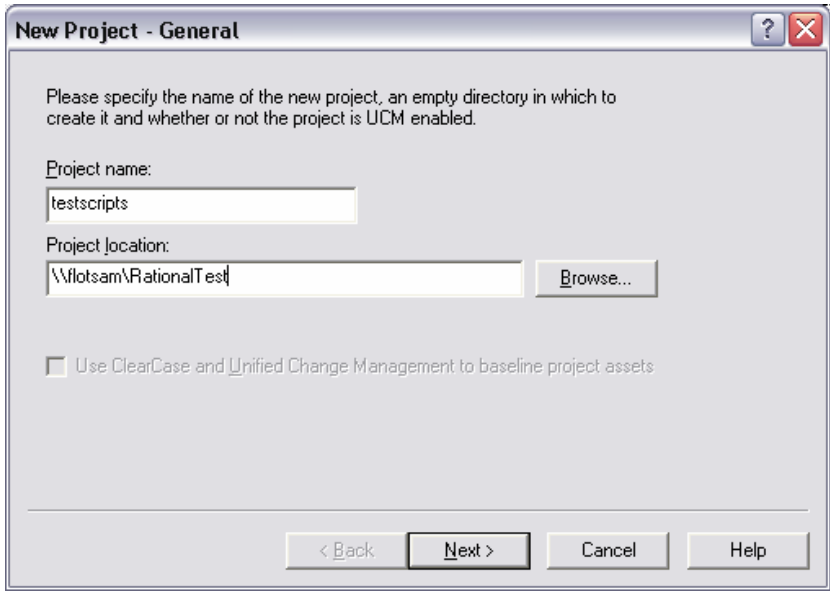

New Project - General
Please specify the name of the new project, an empty directory in which to create it and whether or not the project is UCM enabled.
Project name:
testscripts
Project location:
\\flotsam\RationalTest
Browse...
Use ClearCase and Unified Change Management to baseline project assets
< Back
Next >
Cancel
Help

Figure 9-20 Rational Robot Project

4. Click **Next**. If you do create a password for the Rational project, supply the password on the Security page (see Figure 9-21 on page 341). If you do not create a password, then leave the fields blank on this page.

New Project - Security
Each Rational Project may be associated with a password. This password is required to connect to, configure or delete the project.
Password:
Confirm:
< Back
Next >
Cancel
Help

Figure 9-21 Configuring project password

5. Click **Next** on the Summary page and select **Configure Project Now** (see Figure 9-22 on page 342). The Configure Project dialog box appears (see Figure 9-23 on page 343).

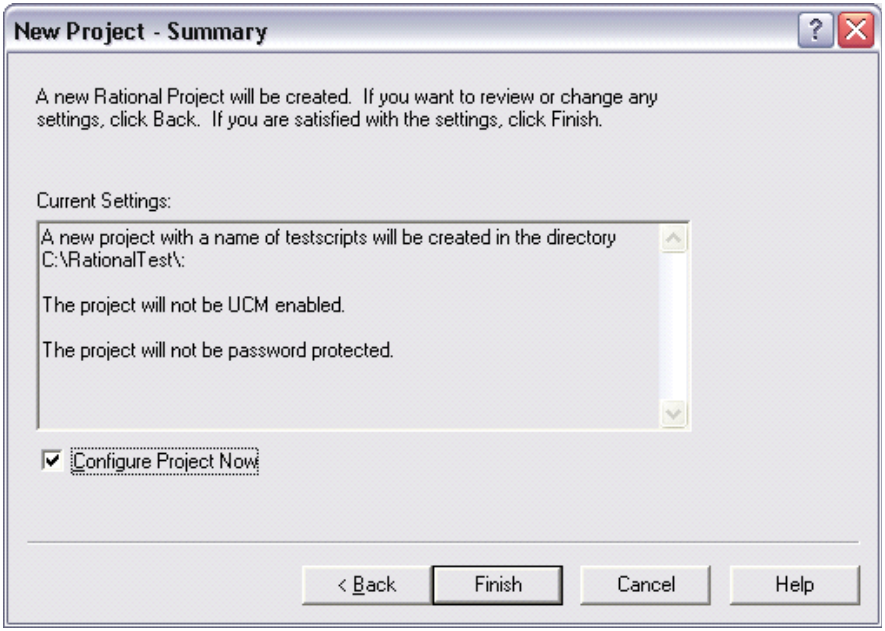

New Project - Summary
A new Rational Project will be created. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Finish.
Current Settings:
A new project with a name of testscripts will be created in the directory C:\RationalTest\:
The project will not be UCM enabled.
The project will not be password protected.
Configure Project Now
< Back
Finish
Cancel
Help

Figure 9-22 Finalize project

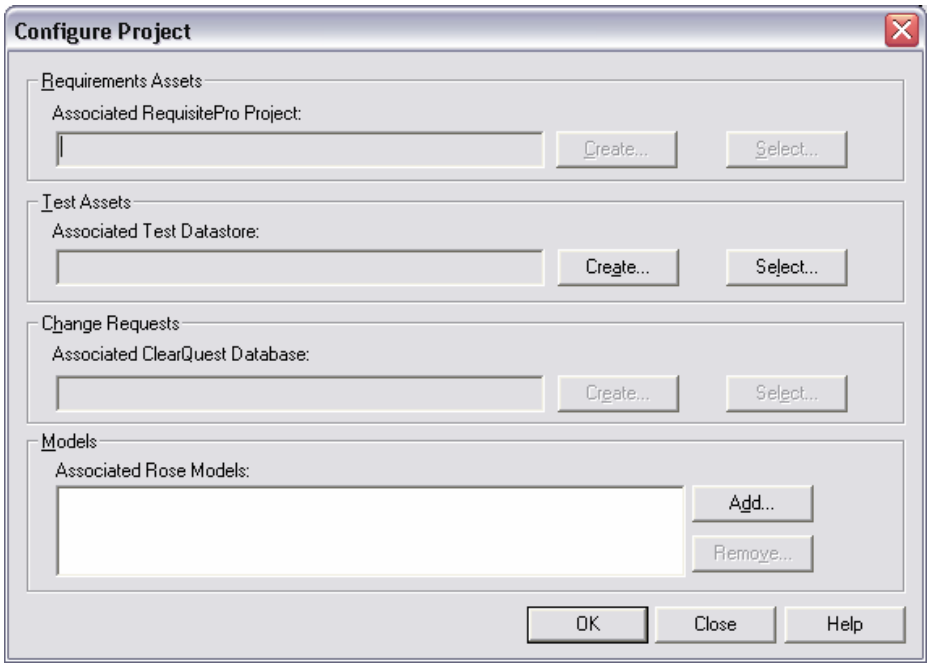


Figure 9-23 Configuring Rational Project

A Rational Test datastore is a collection of related test assets, including test scripts, suites, datapools, logs, reports, test plans, and build information.

You can create a new Test datastore or associate an existing Test datastore.

For testing Rational Robot, the user must set up the Test datastore.

To create a new Test datastore:

1. In the Configure Project dialog box, click **Create** in the Test Assets area. The Create Test Datastore tool appears (see Figure 9-24 on page 344).

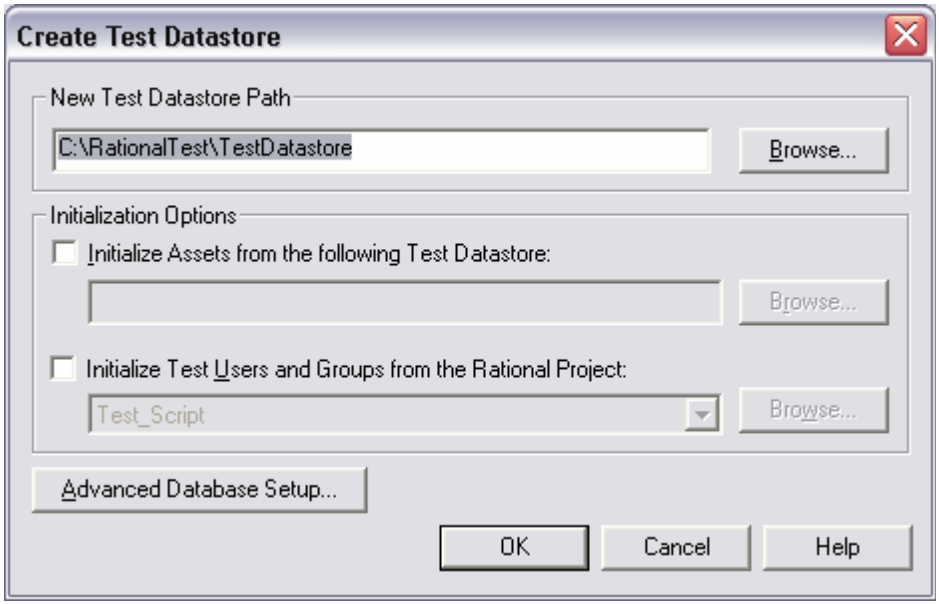


Figure 9-24 Specifying project datastore

2. In the Create Test Datastore dialog box:
   a. In the New Test Datastore Path field, use a UNC path name to specify the area where you would like the tests to reside.
   b. Select initialization options as appropriate.
   c. Click **Advanced Database Setup** and select the type of database engine for the Test datastore.
   d. Click **OK**.

9.1.3 Recording types: GUI and VU scripts

The kind of recordings that can be performed with Rational Robot can be divided in two different types:

- GUI scripts
- VU scripts

GUI scripts are used to record simulations interacting with a graphical application. These scripts are easy to use, but have the drawback of not allowing more than one script to execute at a time, and a requirement to have direct access to the computer desktop screen. On the other hand, they allow for recording very detailed graphical interaction (mouse movements, keystrokes,

and so on) and allow the use of Verification Points to ensure that operations outcomes are those expected. The language used to generate the script is SQABasic, and GUI scripts can be played back with Rational Robot or as part of Rational Test manager.

GUI scripts can be used in a set of complex transactions (repeated continuously) to measure a performance baseline that can be compared when the server configuration changes or to ensure that the end user experience is satisfactory from the end-user point of view (to satisfy an SLA).

VU scripts record the client server requests at the network layer only for specific supported application types and can be used to record outgoing calls performed by the client (network recording) or incoming calls on the server (proxy recording). VU scripts do not support Verification Points and cannot be used to simulate activity on a Generic Windows applications. VU only supports specialized network protocols, not generic API access on the network layer, and VU scripts can only be played back using Rational Test Manager. The playback of VU scripts is not supported by TMTP Version 5.2, so VU will be ignored in this book.

9.1.4 Steps to record a GUI simulation with Rational Robot

There are differences in how a simulation recording is set up and prepared on different applications. For example, to record an HTTP simulation in a browser, you need to load the Extension for the browser you will be using, while with Java, you need to load the Extension and configure the Java Enabler on the JVM you will be using. But whatever application you are using, there are common points that will be followed.

1. Record the script on the GUI.
2. Add features to the script during recording (ARM API calls for TMTP, Verification Points, Timers, comments, and so on).
3. Compile the script.
4. Play the script back for debugging.
5. Save and package the script for TMTP Version 5.2.

Record the script on the GUI

To record a GUI script, click the Record GUI script on the toolbar:

Type an application name in the Record GUI Dialog (Figure 9-25).

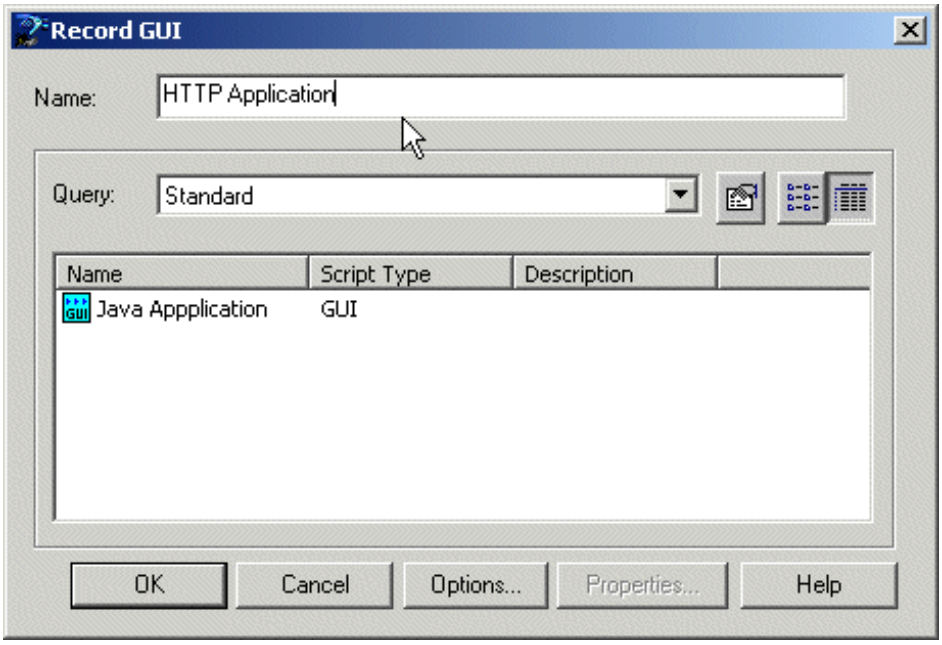


Figure 9-25 Record GUI Dialog Box

Click on **OK**, and Rational Robot will minimize while the Recording toolbar is displayed:

The Recording toolbar contains the following buttons: Pause the recording, Stop the recording, Open the Rational Robot main window, and Display the GUI Insert toolbar. The first three are self-explanatory; the last is needed to easily add features to the script being recorded using the GUI Insert toolbar (Figure 9-26).

Figure 9-26 GUI Insert

From this toolbar you can add Verification Points, start the browser on a Web page for recording, and so on.

Add Verification Points to the script

During the GUI simulation flow, it is a good idea to insert Verification Points, which are points in the program flow that save information on GUI objects for comparing with the expected state. When you create a Verification Point, you select a Verification Method (case sensitivity, sub string, numeric equivalence, numeric range, or blank field) and an Identification Method (by content, location, title, and so on); with Verification Points, you can also insert timers and timeouts in the program flow. Verification is especially needed to ensure that if the application has delays in the execution, Rational Robot will wait for the Verification Point to pass before continuing the execution.

Verification Points can be created on Window Regions and Window Images using OCR, but in the case of e-business applications, Object Properties Verification Points are easier to use, reliable, and less subject to suffer changes in the application interface or data displayed.

The state of an application working in a client server environment during the playback of a simulation often changes if the data retrieved from the server is different from the one retrieved during the recording, so to avoid errors during the playback, it is a good idea to use Verification Points. Using Verification Points, you can verify that an object's properties are those expected.

Verification Points can be added in a script:

1. During the recording
2. While editing the script after the recording

In both cases, you need to press the Display GUI insert Toolbar in the Rational Robot floating toolbar during the recording or on the Standard Toolbar while editing, but you must be sure that the cursor is at the point where you want to add the Verification Point if you have already recorded the script. After you press the Display GUI Insert Toolbar button, you will see the GUI Insert toolbar floating (Figure 9-26 on page 346).

Select the type of Verification Point needed, for example, Object Properties, and type a name for the Verification Point in the Verification Point Name dialog (Figure 9-27 on page 348).

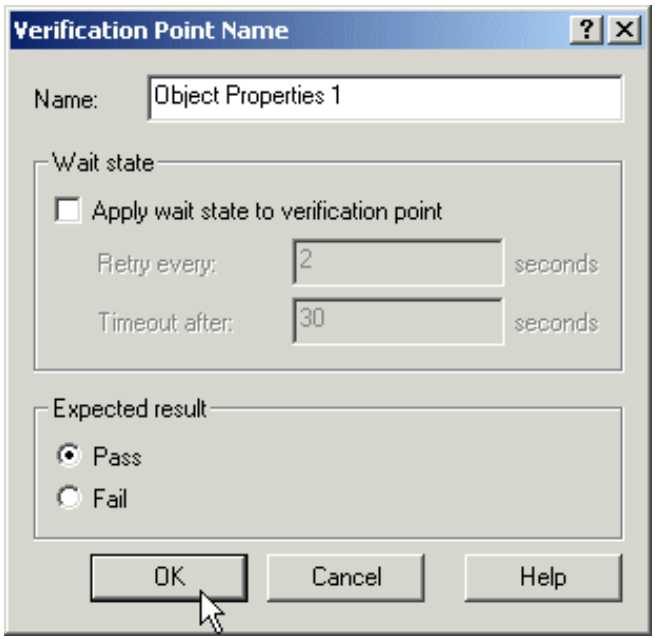


Figure 9-27 Verification Point Name Dialog

In case the object you will use as a Verification Point takes some time to be displayed or to get to the desired state, check the **Apply wait state to Verification Point** check box and select the retry and time-out time in seconds. Also, select the desired state; in simulations, you generally always expect the result to be of Pass type. Click on **OK** when you complete all the settings, and the Object Finder dialog is displayed, as in Figure 9-28 on page 349.

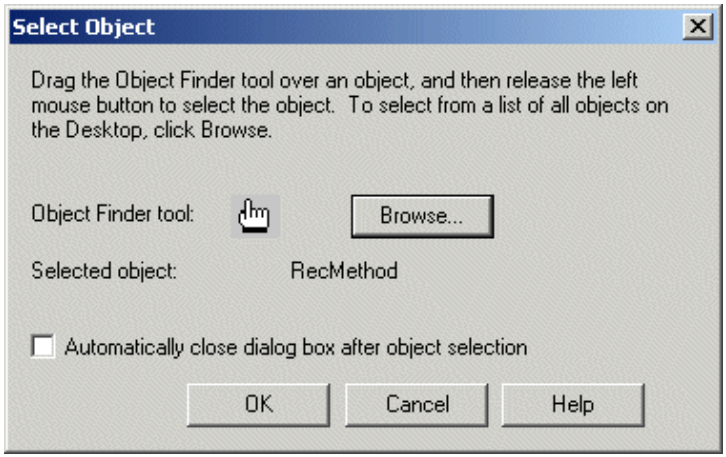

Select Object
Drag the Object Finder tool over an object, and then release the left mouse button to select the object. To select from a list of all objects on the Desktop, click Browse.
Object Finder tool:
Browse...
Selected object:
RecMethod
Automatically close dialog box after object selection
OK
Cancel
Help

Figure 9-28 Object Finder Dialog

Select the icon of the Object Finder tool and drag it on the object whose properties you want to investigate. A flyover appearing on each object will tell you how it is identified, for example, a Java label will show a tool tip showing a Java label when the Object Finder tool is on it. When the mouse is released, the properties for the object you selected are displayed in the Object Properties Verification Point panel (Figure 9-29 on page 350).

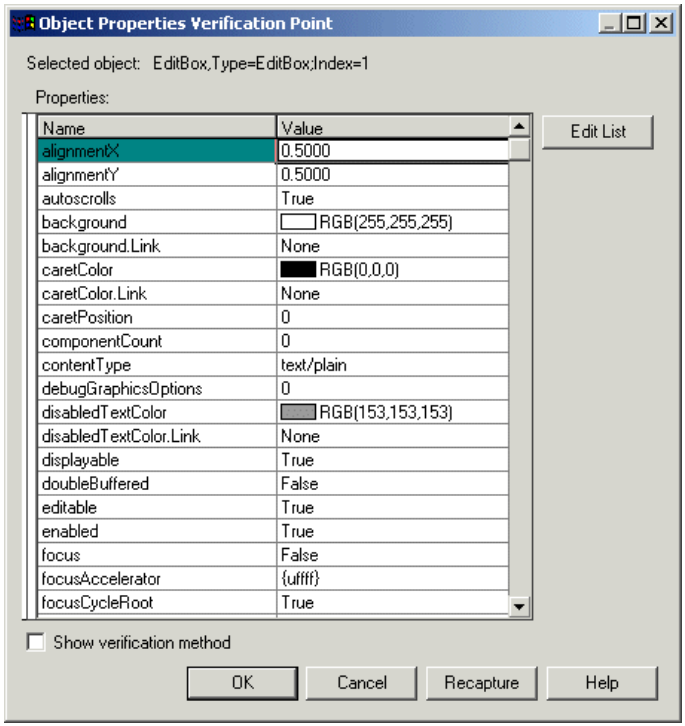


Figure 9-29 Object Properties Verification Point panel

Select the property/value pair that you want to check in the Verification Point and click on **OK**.

If you were recording the simulation, the Verification Point will be included in the correct point of the script. If you where adding the Verification Point after the script recording, the Verification Point will be included where the cursor was in the script.

Here is how a Verification Point on a Java Label would look like in the script (Example 9-1).

Example 9-1 Java Label Verification Point

```
Result = LabelVP (CompareProperties, "Type=Label;Name=TryIt Logo", "VP=Object
Properties;Wait=2,30")
```

Add timers to the script

Rational Robot supports the use of timers in scripts to measure performance, but these timers do not support the ARM API standard and cannot be used to

measure transaction performance with TMTP. Timers are inserted using the Start Timer button in the GUI Insert Toolbar, but you will also need to add ARM API calls to the script to capture transaction performance.

Timers can still be valuable to use if you want to have an idea of how long a transaction takes on the fly; in this case, you can insert timers together with ARM API calls.

Use comments in the script for maintenance

It is a good idea to record comments in the script during execution, in particular where you pressed particular code sequences or typed down information that was relevant only in that particular step. For example, suppose you are testing a Web-based interface that pulls information from a database. Since the information retrieved can change over time while the interface of the application does not, when you add a Verification Point on a graph that is dynamically generated, add a comment to remind you that the portion of script may need further coding.

9.1.5 Add ARM API calls for TMTP in the script

ARM API calls need to be included in the script by manually editing the code; the instructions you add will load the ARM function library, define ARM return codes for use in the script, will initialize the simulation so that ARM will consider the API calls coming from it, and define the start and stop points for each transaction.

You can create any number of transactions inside the script sequentially, including one in another or overlapping.

To load the ARM API, you can add code similar to Example 9-2 in the script header, or cut the sample below and paste it into your script directly. This may help avoid typing errors.

Example 9-2 Script ARM API declaration

```
Declare Function arm_init Lib "libarm32"(ByVal appl_name As String,ByVal
appl_userid As String,ByVal flags As Long,ByVal data As String,ByVal data_size
As Long)As Long

Declare Function arm_getid Lib "libarm32"(ByVal appl_id As Long,ByVal tran_name
As String,ByVal tran_detail As String,ByVal flags As Long,ByVal data As
String,ByVal data_size As Long)As Long

Declare Function arm_start Lib "libarm32"(ByVal tran_id As Long,ByVal flags As
Long,ByVal data As String,ByVal data_size As Long)As Long
```

```
Declare Function arm_stop Lib "libarm32"(ByVal start_handle As Long,ByVal
tran_status As Long,ByVal flags As Long,ByVal data As String,ByVal data_size As
Long)As Long

Declare Function arm_end Lib "libarm32"(ByVal appl_id As Long,ByVal flags As
Long,ByVal data As String,ByVal data_size As Long)As Long
```

To declare variables to hold returns from ARM API calls, add the script in Example 9-3.

Example 9-3 ARM API Variables

```
Dim appl_handle As Long
Dim getid_handle As Long
Dim start_handle As Long
Dim stop_rc As Long
Dim end_rc As Long
```

All the code above can be put at the top of the script. Next, you must initialize the simulation as an ARM'ed application, and to do this, you perform the operations shown in Example 9-4 in the script.

Example 9-4 Initializing the ARM application handle

```
appl_handle = arm_init("GenWin","*",0,"0",0)
```

The code in Example 9-4 retrieves an application handle using the ARM API so that the application is universally defined; this is needed because with applications that have been ARM instrumented in the source code, you might have multiple instances of the same application running at a time.

Important: In order for the TMTP Version 5.2 GenWin component to be able to retrieve the ARM data generated with this Rational Robot script, the Application handle needs to use the value “GenWin”, as shown in Example 9-4.

Next, you need a transaction identifier, and you will need one for each transaction your script will simulate.

Important: The second parameter should match the pattern “ScriptName.*”, where the .* indicates any characters, and ScriptName is the name of the Rational Robot Script. Using our example above, valid transaction IDs could be “MyTransaction” and “MyTransactionSubtransaction1”. The third parameter is the description, which will be displayed in the TMTP Topology view, so it should be a value that will provide useful information when viewing the Topology.

As you can see, the application handle is sent to the ARM API and a transaction handle is retrieved (Example 9-5).

Example 9-5 Retrieving the transaction handle

```
getid_handle =a rm_getid(appl_handle,"MyTransaction","LegacySystemTx",0,"0",0)
```

Now you can start the transaction. The line below (Example 9-6) needs to precede the script steps where the transaction you want to measure takes place.

Example 9-6 Specifying the transaction start

```
start_handle =arm_start(getid_handle,0,"0",0)
```

Again, ARM gets a handle ad returns another; in this case, it gets the transaction handle you got and returns a start handle. The handle is needed to end the right transaction.

After the transaction completes with a successful Verification Point, you need to end the transaction using the call in Example 9-7.

Example 9-7 Specifying the transaction stop

```
stop_rc = arm_stop(start_handle,0,0,"0",0)
```

This will close the transaction. As you can see, we ensure that we are closing the transaction by starting the stop call, which includes the transaction start handle.

The last call (Example 9-8) you need is for cleanup purposes and can be included at the end of the script. The end call sends the application handle you received with the initialization.

Example 9-8 ARM cleanup

```
end_rc = arm_end(appl_handle,0,"0",0)
```

This will complete the set of API calls for the transaction you are simulating.

Compile the script

Rational Robot scripts are compiled before playback begins. The compilation can be started by the user by clicking on the **Compile** button to ensure that the script is formally correct, or the compile stage can be left to Rational Robot that takes care of it whenever a change is done to the source.

Scripts are recorded with the rec extension; their compiled form is sbx. Include files have the sbh extension and are automatically compiled by Rational Robot, so the user does not have to worry about them in any case.

Debugging scripts

Rational Robot includes a fully functional debugging environment you can use to ensure that your script flow is correct and that all edge cases are covered during the execution.

Starting the debugging process also compiles the script in case it has just been recorded or if the source has been changed.

To start debugging, open an existing script or record a new script and click on the Debug menu. The menu is displayed, as shown in Figure 9-30.

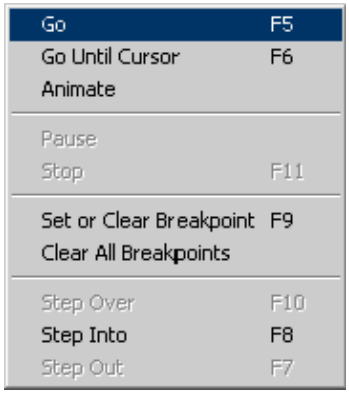


Figure 9-30 Debug menu

Before starting to debug, you will probably need to set breakpoints in the script to run a portion of script that is already working. To use breakpoints, move the cursor in the script up to where the breakpoint to be set and select **Set or Clear Breakpoint** to set or clear a breakpoint at that point in the script. You can also simply press F9 to set or clear breakpoints on the current line.

To run the script up to the selected line, you have to select **Go Until Cursor in the Debug** menu or press F6; this will start playback of the script and stop before executing the line that is currently selected. At any time, you can choose the Step

Over, Step Into, and Step Out buttons, which work as in any other debugging environment.

One interesting option you have in the Debug menu is the Animate option; this will play back the script in Animation Mode. Animation Mode plays the script by highlighting, in yellow, each line that is executed. Keep in mind that the script will still playback at considerable speed, not giving you time to evaluate what is occurring; it is a good idea to increase the delay between key strokes to ensure that you can analyze the execution flow. To do this, you can change the delay between commands and keystrokes by selecting **Tools** → **GUI Playback Options**. This will display the GUI Playback Options dialog (Figure 9-31).

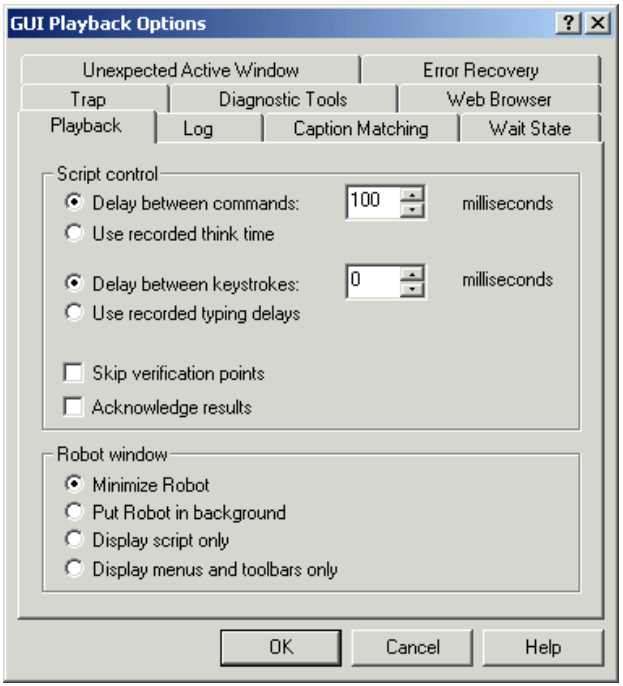


Figure 9-31 GUI Playback Options

Select the **Playback** tab and increase the **Delay between commands** to 2000; this will leave a two second delay between commands during the playback. You can also increase the **Delay between keystrokes** to 100 if you want better visual control on the keys being pressed. Click on **OK** when you are done and get back to the script. The next time you select Animate in the Debug menu, you will have more time to understand what the script is doing.

If the machine used to record and debug the simulation is the same that will execute, ensure that you set **Delay between commands** back to 100 and **Delay between keystrokes** back to 0 before playing back the script with TMTP.

Other than executing scripts to a specific line and running in Animation Mode, you can also investigate variable values in the Variable window. This window is not enabled by default; to ensure that you see it, you must select **Variables** in the View menu. The Variable window will be displayed in the right-lower corner of the Rational Robot window, but can be moved around the main window and docked where you prefer.

The values you see in this window are updated at each step of script playback.

Other interesting items

Other than those mentioned above, Rational Robot includes a set of extra features that you might be interested in. For example, you can use datapools to feed data in the simulation that will change data entered in specific fields, or use an Authentication Datapool if you want to store passwords and login IDs separately from the script (although we recommend encrypting passwords locally using VB code; the following section , “Obfuscating embedded passwords in Rational Scripts” on page 356 describes how to do this). You may also be interested in tips regarding screen locking discused in , “Rational Robot screen locking solution” on page 360.

Obfuscating embedded passwords in Rational Scripts

Often, when recording Rational Scripts, it is necessary to record user IDs and passwords. This has the obvious security exposure that if your script is viewed, the password will be viewable in clear text. This section describes a mechanism for obfuscating the password in the script.

This mechanism relies on the use of an encryption library. The encryption library that we used is available on the redbook Web site. The exact link can be found in Appendix C, “Additional material” on page 473.

First, the encryption library must be registered with the operating system. For our encryption library, this was achieved by running the command:

```
regsvr32.exe EncryptionAlgorithms.dll
```

Once you have run this command, you must encrypt your password to a file for later use in your Rational Robot scripts. This can be achieved by creating a Rational Robot Script from the text in Example 9-9 on page 357 and then running the resulting script.

Example 9-9 Stashing obfuscated password to file

```
Sub Main

    Dim Result As Integer
    Dim bf As Object
    Dim answer As Integer

    ' Create the Encryption Engine and store a key
    Set bf = CreateObject("EncryptionAlgorithms.BlowFish")
    bf.key = "ibm"

    Begin Dialog UserDialog 180, 90, "Password Encryption"

        Text 10, 10, 100, 13, "Password: ", .lblPwd
        Text 10, 50, 100, 13, "Filename: ", .lblFile
        TextBox 10, 20, 100, 13, .txtPwd
        TextBox 10, 60, 100, 13, .txtFile

        OKButton  131, 8, 42, 13
        CancelButton  131, 27, 42, 13

    End Dialog

    Dim myDialog As UserDialog

DialogErr:
    answer = Dialog(myDialog)
    If answer <> -1 Then
        Exit Sub
    End If

    If Len(myDialog.txtPwd) < 3 then
        MsgBox "Password must have more than 3 characters!", 64, "Password
          Encryption"
        GoTo DialogErr
    End If

    ' Encrypt
    strEncrypt = bf.EncryptString(myDialog.txtPwd, "rational")


    ' Save to file
    'Open "C:\secure.txt" For Output Access Write As #1
    'Write #1, strEncrypt
    Open myDialog.txtFile For Output As #1
    If Err <> 0 Then
        MsgBox "Cannot create file", 64, "Password Encryption"
        GoTo DialogErr
```

```
    End If

    Print #1, strEncrypt
    Close #1

    If Err <> 0 Then
        MsgBox "An Error occurred while storing the encrypted password", 64,
          "Password Encryption"
        GoTo DialogErr
    End If

    MsgBox "Password successfully stored!", 64, "Password Encryption"

End Sub
```

Running this script will generate the pop-up window shown in Figure 9-32, which asks for the password and name of a file to store the encrypted version of that password within.

Figure 9-32 Entering the password for use in Rational Scripts

Once this script has run, the file you specified above will contain an encrypted version of your password. The password may be retrieved within your Rational Script, as shown in Example 9-10.

Example 9-10 Retrieving the password

```
Sub Main
    Dim Result As Integer
    Dim bf As Object
    Dim strPasswd As String
    Dim fchar()
    Dim x As Integer

    ' Create the Encryption Engine and store a key
```

```
    Set bf = CreateObject("EncryptionAlgorithms.BlowFish")
    bf.key = "ibm"

    ' Open file and read encrypted password
    Open "C:\encryptedpassword.txt" For Input Access Read As #1
    Redim fchar(Lof(1))
    For x = 1 to Lof(1)-2
        fchar(x) = Input (1, #1)
        strPasswd = strPasswd & fchar(x)
    Next x

    ' Decrypt
    strPasswd = bf.DecryptString(strPasswd, "rational")
    SQAConsoleWrite "Decrypt: " & strPasswd


End Sub
```

The resulting unencrypted password has been retrieved from the encrypted file (in our case, we used the encryptedpassword.txt file) and placed into the variable strPasswd, and the variable may be used in place of the password where required. A complete example of how this may be used in a Rational Script is shown in Example 9-11.

Example 9-11 Using the retrieved password

```
Sub Main

    'Initially Recorded: 10/1/2003  11:18:08 AM
    'Script Name: TestEncryptedPassword

    Dim Result As Integer
    Dim bf As Object
    Dim strPasswd As String
    Dim fchar()
    Dim x As Integer

    ' Create the Encryption Engine and store a key
    Set bf = CreateObject("EncryptionAlgorithms.BlowFish")
    bf.key = "ibm"

    ' Open file and read encrypted password
    Open "C:\encryptedpassword.txt" For Input Access Read As #1
    Redim fchar(Lof(1))
    For x = 1 to Lof(1)-2
        fchar(x) = Input (1, #1)
        strPasswd = strPasswd & fchar(x)
    Next x
```

```
' Decrypt the password into variable
strPasswd = bf.DecryptString(strPasswd, "rational")

Window SetContext, "Caption=Program Manager", ""
ListView DblClick, "ObjectIndex=1;\;ItemText=Internet Explorer",
      "Coords=20,30"

Window SetContext, "Caption=IBM Intranet - Microsoft Internet Explorer", ""
ComboEditBox Click, "ObjectIndex=2", "Coords=61,5"
InputKeys "http://9.3.4.230:9082/tmtpUI{ENTER}"
InputKeys "root{TAB}^+{LEFT}"

' use the un-encrypted password retrieved from the encrypted file.
InputKeys strPasswd

PushButton Click, "HTMLText=Log On"
Toolbar Click, "ObjectIndex=4;\;ItemID=32768", "Coords=20,5"
PopupMenuSelect "Close"


End Sub
```

Rational Robot screen locking solution

Some users of TMTP have expressed a desire to be able to lock the screen while the Rational Robot is playing. The best and most secure solution to this problem is to lock the endpoint running simulations in a secure cabinet. There is no easy alternative solution, as the Rational Robot requires access to the screen context while it is playing back. During the writing of this redbook, we attempted a number of mechanisms to achieve this result, including use of Windows XP Switch User functionality, without success. The following Terminal Server solution implemented at one IBM customer site was suggested to us. We were unable to verify it ourselves, but we considered it useful information to provide as a potential solution to this problem.

This solution relies on the use of Windows Terminal Server, which is shipped with the Windows 2000 Server. When a user runs an application on Terminal Server, the application execution takes place on the server, and only the keyboard, mouse, and display information is transmitted over the network. This solution relies on running a Terminal Server Session back to the same machine and running the Rational Robot within the Terminal Server session. This allows the screen to be locked and the simulation to continue running.

1. Ensure that the Windows Terminal Server component is installed. If it is not, it can be obtained from the Windows 2000 Server installation CD from the Add On components dialog box (see Figure 9-33 on page 361).

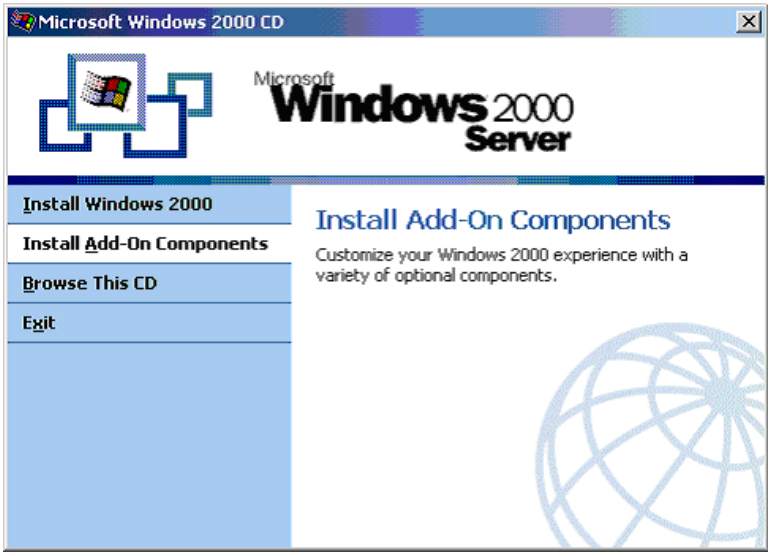


Figure 9-33 Terminal Server Add-On Component

As the Terminal Server session will be back on the local machine, there is no reason to install the Terminal Server Licensing feature. Due to this fact, you should also select the Remote Administration mode option during Terminal Server install.

After the Terminal Server component is installed, you will need to reboot your machine.

2. Install the Terminal Server client on the local machine. The Terminal Server install provides a facility to create client installation diskettes. This same source can be used to install the Terminal Server client locally (Figure 9-34 on page 362) by running the setup.exe (the path to this setup.exe is, by default, c:\winnt\system32\clients\tsclient\win32\disks\disk1).

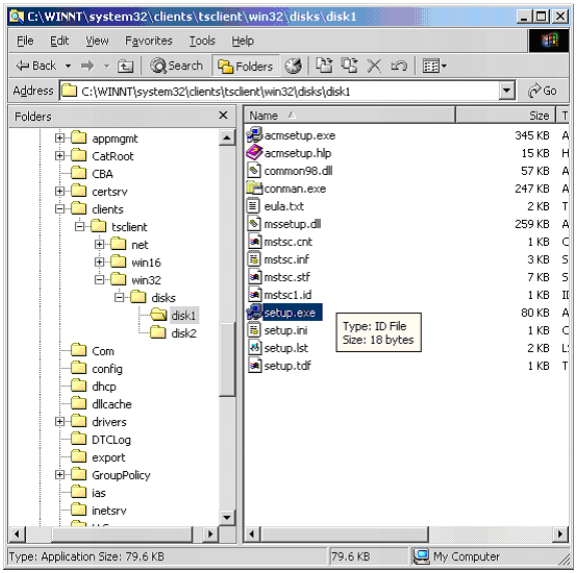


Figure 9-34 Setup for Terminal Server client

3. Once you have installed the client, you may start a client session from the appropriate menu option. You will be presented with the dialog shown in Figure 9-35 on page 363. From this dialog, you should select the local machine as the server you wish to connect to.

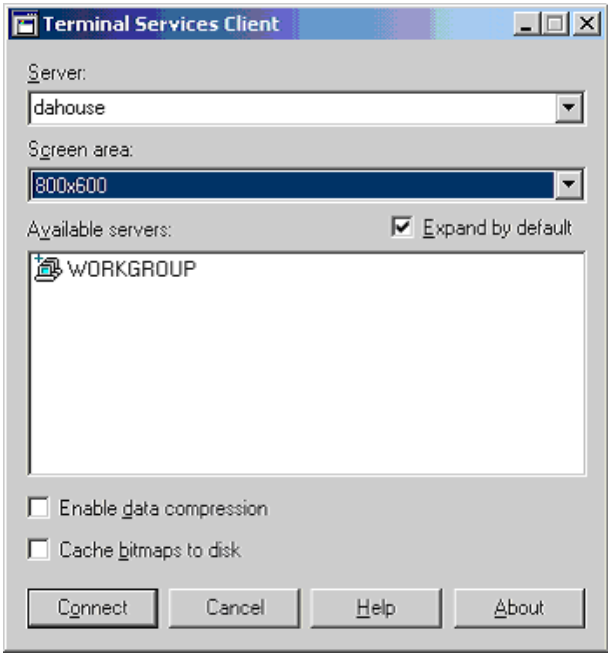


Figure 9-35 Terminal Client connection dialog

> **Note:** It is useful to set the resolution to one lower than that used by the workstation you are connecting from. This allows the full Terminal Client session to be seen from the workstation screen.

4. Once you have connected, you will be presented with a standard Windows 2000 logon screen for the local machine within your client session. Log on as normal.
5. Now you can run your Rational Robot scripts using whichever method you would normally do this, with the exception of via GenWin. You may now lock the host screen and the Rational Robot will continue to run in the client session.

Recording a GUI simulation on an HTTP application

There is an important difference you must consider when you start to record a simulation on a browser-based application: the browser window must be started by Rational Robot. You should not click on the Record GUI script and then start the browser by clicking on a Desktop link.

To record the GUI simulation, do the following steps:

1. Click on the Display GUI Insert toolbar button located in the GUI Record toolbar:

This displays the GUI Insert toolbar:

2. Click on the Start browser button:

This will display the Start browser dialog (Figure 9-36), where you must type down the initial address the browser has to start with and a Tag that will be used by Rational Robot to identify the correct browser window if there are multiple windows running.

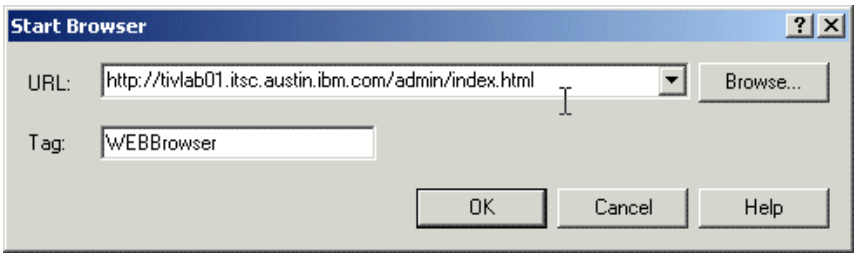


Figure 9-36 Start Browser Dialog

When you click on **OK**, the browser opens on the address specified and all actions performed in the browser are recorded in the script. Apart from the differences to start the application/browser, there are not any major differences compared to the procedure you usually follow for recording any other application simulation.

Recording a GUI simulation on a Java application

Before recording a simulation running on Java, ensure that you installed and configured the Rational Java Enabler on the JVM you will be using and load the Java Extension.

To record a GUI simulation on a Java application, select the Record GUI Script button on the toolbar in the main Rational Robot window and start the application in the usual way.

Simulate and perform all the actions that you need; Rational Robot will record the simulation while you execute, as on any other kind of application.

There are not any differences between Java simulations and generic Windows applications; only the object properties slightly change.

9.2 Introducing GenWin

The GenWin allows centralized management of distributed playback of your Rational Robot Scripts. When you use Rational Robot and Generic Windows together, it allows you to measure how users might experience a Windows application in your environment.

9.2.1 Deploying the Generic Windows Component

In order to play back a Rational Robot script, the Management Agent you intend to use for playback must have the Rational Robot installed and it must have the Generic Windows component installed on it. The procedure for deploying the Rational Robot is covered in 9.1.1, “Installing and configuring the Rational Robot” on page 326. The procedure for deploying the Generic Windows component is outlined below.

1. Select the **Work with Agents** option from the System Administration menu of the Navigation pane. The window shown in Figure 9-37 on page 366 should appear.

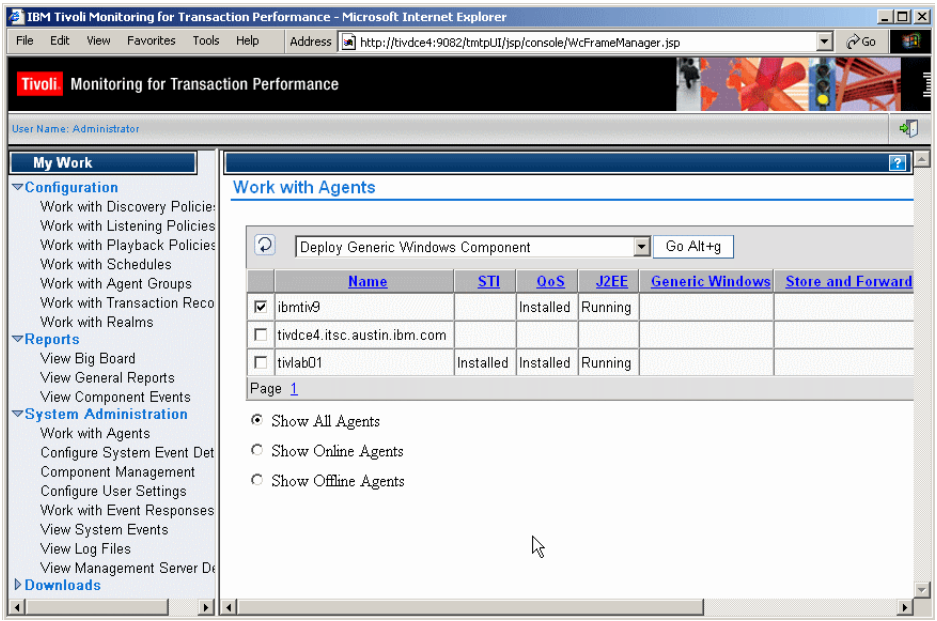


Figure 9-37 Deploy Generic Windows Component

2. Select the Management Agent you wish to deploy the Generic Windows component to from the Work with Agents window.
3. Then select the **Deploy Generic Windows Component** from the drop-down box and press **Go**.
4. This will display the Deploy Components and/or Monitoring Component window (see Figure 9-38 on page 367). In this window, you must enter details about the Rational Robot Project in which your playback scripts are going to be stored.

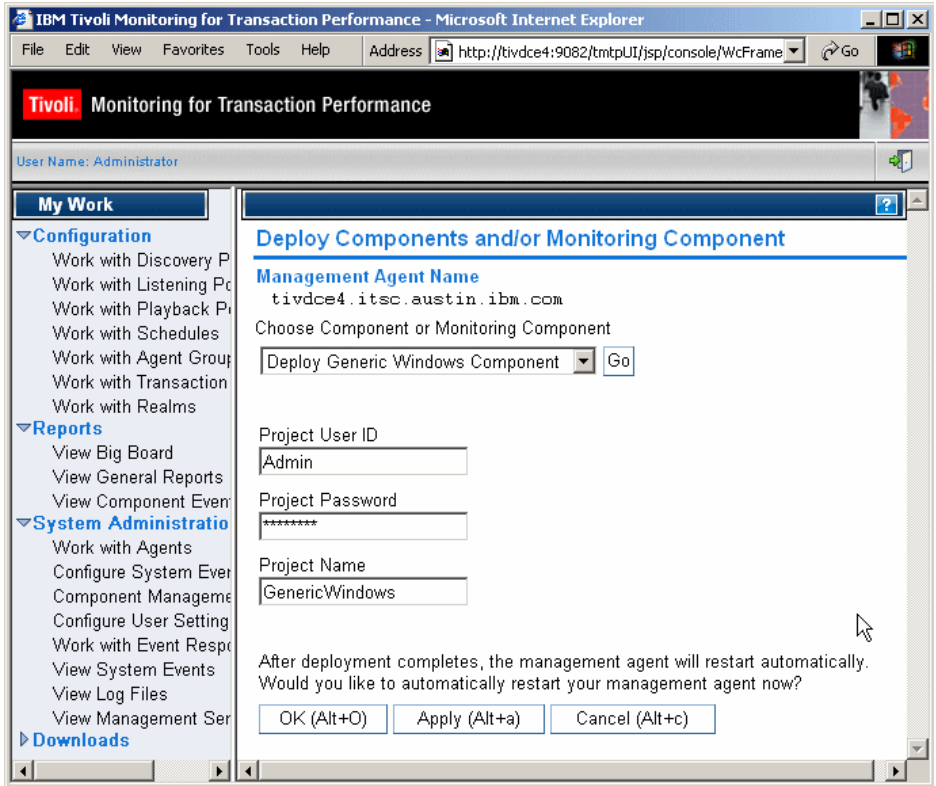


Figure 9-38 Deploy Components and/or Monitoring Component

Tip: The Rational Project does not have to exist prior to this step. In fact, it is far easier to create this Rational Project after deploying the GenWin project, because the Project must be located in the directory $MA\app\genwin\<project> ($MA is the home directory for the Management Agent), and this path is not created until the Generic Windows component has been deployed. After you have deployed the Generic Windows component, you must create a new Rational Robot Project on the Management Agent with details that match the details you have entered into the Deploy Components and/or Monitoring Component window. When you specify playback policies, the Rational Robot scripts will automatically be placed into this project.

5. Create a Rational Robot Project for use by the Generic Windows component for playback. The procedure for creating a Rational Robot Project is covered in 9.1.2, “Configuring a Rational Project” on page 339. In order for GenWin to use the project, it needs to be located using a subdirectory to the

$MA\app\genwin directory. When the project has been created, it will resicde in a subdirectory of the $MA\app\genwin\<project> directory.

9.2.2 Registering your Rational Robot Transaction

Once the Generic Windows component has been deployed, you can register your Rational Robot transaction scripts with TMTP as follows:

1. Select the Work with Transaction Recordings option from the Configuration menu of the Navigation pane. The window shown in Figure 9-39 should appear.

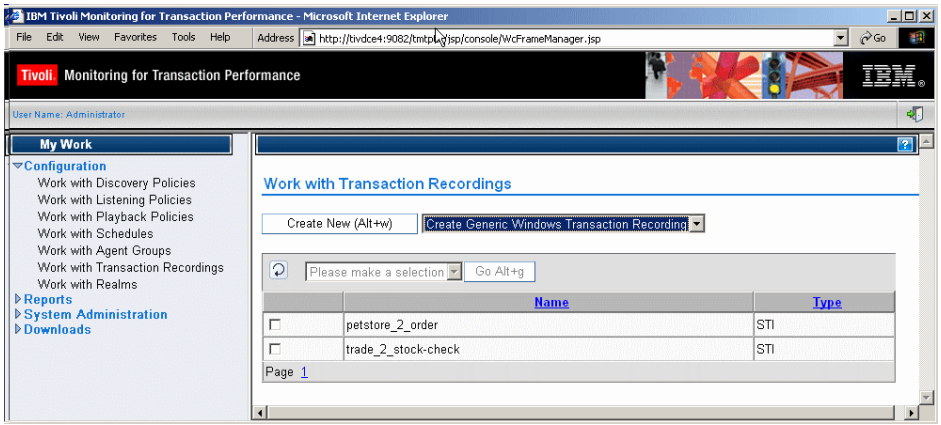


Figure 9-39 Work with Transaction Recordings

2. Select **Create Generic Windows Transaction Recording** from the Create New drop-down box and then push the **Create New** button.
3. In the Create Generic Windows Transaction window (Figure 9-40 on page 369), which you are now presented with, you need to provide the Rational Robot Script files. This can be done using the **Browse** button.

Tip: It is easier to add the two script files required in the Create Generic Windows Transaction window if you are running your TMTP browser from the machine on which the scripts are located. By default, these two files will be located in the $ProjectDir\TestDataStore\DefaultTestScriptDataStore\TMS_Scripts directory ($ProjectDir is the directory in which your source Rational Robot project is located).

Two files are required for each recording: a .rec, and a .rtxml file. For example, if the script you recorded was named TestNotepad, you would need

to add both the TestNotepad.Script.rtxml and TestNotepad.rec files. Once you have added both files, press the **OK** button.

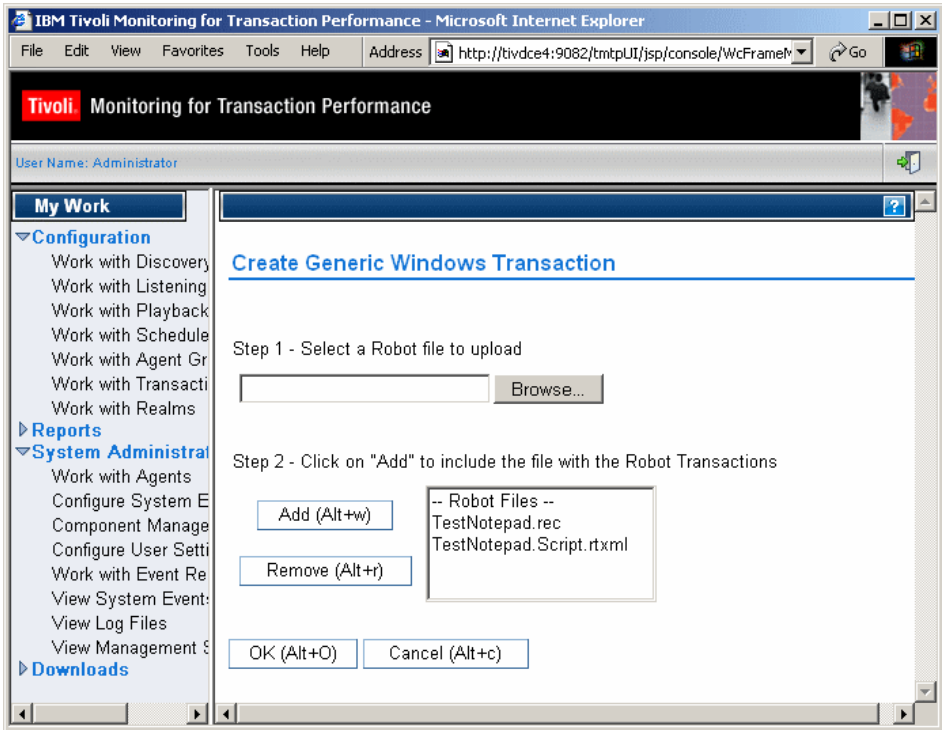


Figure 9-40 Create Generic Windows Transaction

9.2.3 Create a GenWin playback policy

Now that you have registered a Rational Robot Transaction with TMTP, you can specify how you wish to play the transaction back by creating a Playback Policy. The procedure for deploying the Generic Windows component is outlined below.

1. Select the **Work with Playback Policies** option from the Configuration menu of the Navigation pane.
2. Select **Generic Windows** from the Create New drop down box and then press the **Create New** button (see Figure 9-41 on page 370).

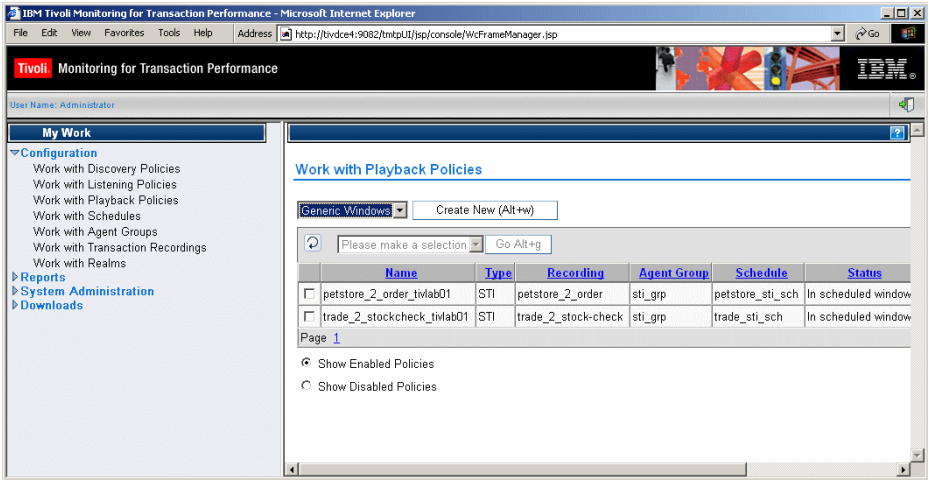


Figure 9-41 Work with Playback Policies

You are then presented with the Create Playback Policy workflow (see Figure 9-42).

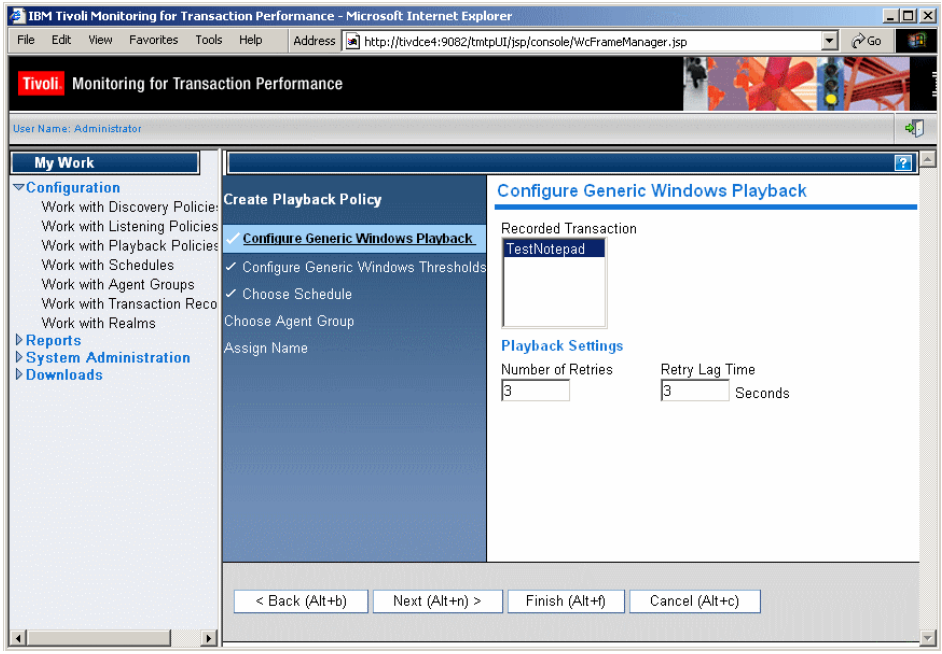


Figure 9-42 Configure Generic Windows Playback

3. Configure the Generic Windows playback options. From here you can select the transaction that you have previously registered. You can also configure the number or retries and amount of time between each retry (if you specify three retries, the transaction will be attempted four times). Once you are happy with the settings, press the **Next** button.
4. The next part of the workflow allows you to configure the Generic Windows thresholds (see Figure 9-43). This allows you to set both performance and availability thresholds, as well as associating Event Responses with those thresholds (for example, running a script, generating an Event to TEC, generating an SNMP Trap, or sending an e-mail). By default, Events are only generated and displayed in the Component Event view (accessed by selecting **View Component Events** from the Reports menu in the Navigation area).

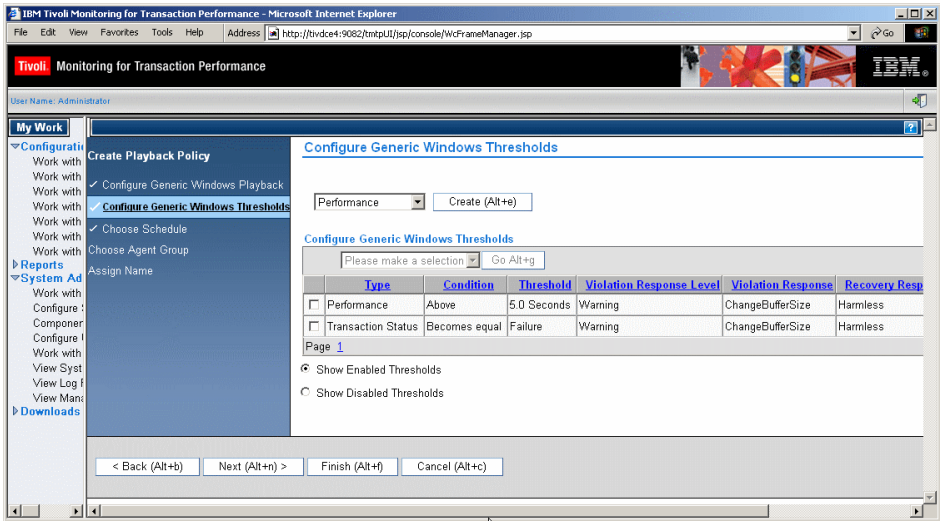


Figure 9-43 Configure Generic Windows Thresholds

Note: If you are unsure what thresholds to set, you may take advantage of TMTP's automatic baseline and thresholding mechanism. This is explained in 8.3, "Deployment, configuration, and ARM data collection" on page 239.

5. Configure the schedule you wish to use to playback the Rational Robot script (see Figure 9-44 on page 372). You may use schedules you have previously created or create a new one.

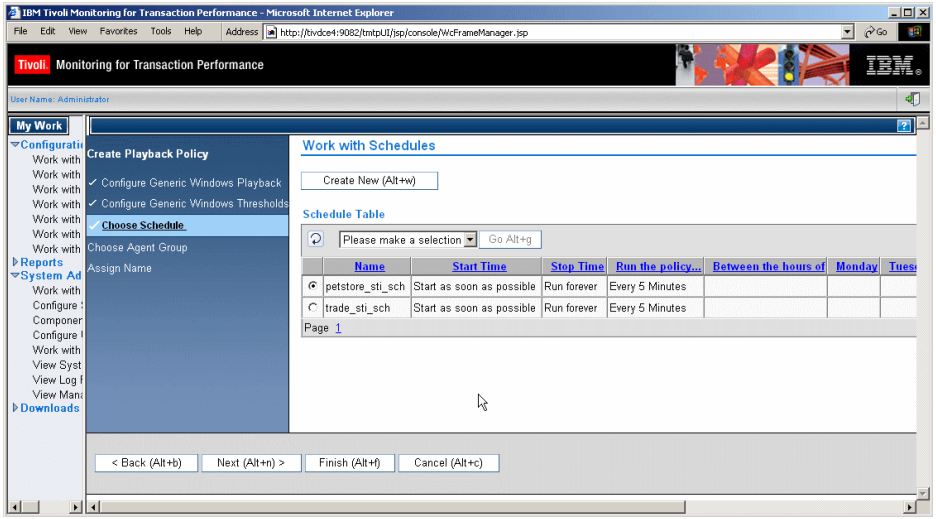


Figure 9-44 Choosing a schedule

Note: The Rational Robot has a practical limit to the number of transactions that can be played back in a given period. During our experiments, we found each invocation of the Robot at the Management Agent took 30 seconds to initialize prior to playing the recording. This meant that it was only possible to play back two transactions a minute. There are several ways in which this shortcoming could be overcome. One way is to use a Rational Robot Script that includes more than one transaction (for example, loops over the one transaction many times within the one script). Another mechanism may be the use of multiple virtual machines on the one host, with each virtual machine hosting its own Management Agent.

6. Choose an agent group on which you want to run the playback (see Figure 9-45 on page 373). Each of the Management Agents in the agent group must have had the Generic Windows component installed on it and the associated Rational Robot project created.

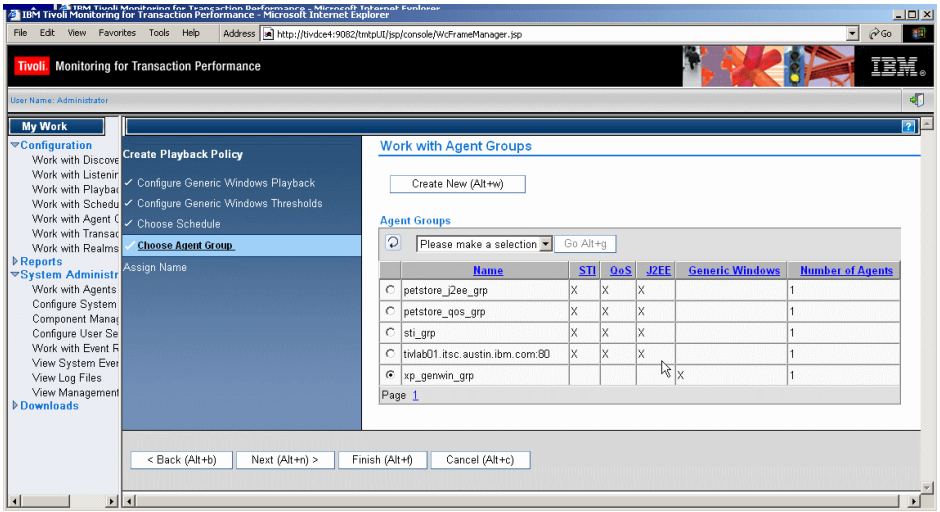


Figure 9-45 Specify Agent Group

7. Give the Playback Policy a name, description, and specify if you want the policy pushed out to the agents immediately or at the next polling interval (by default, polling intervals are every 15 minutes) (see Figure 9-46 on page 374).

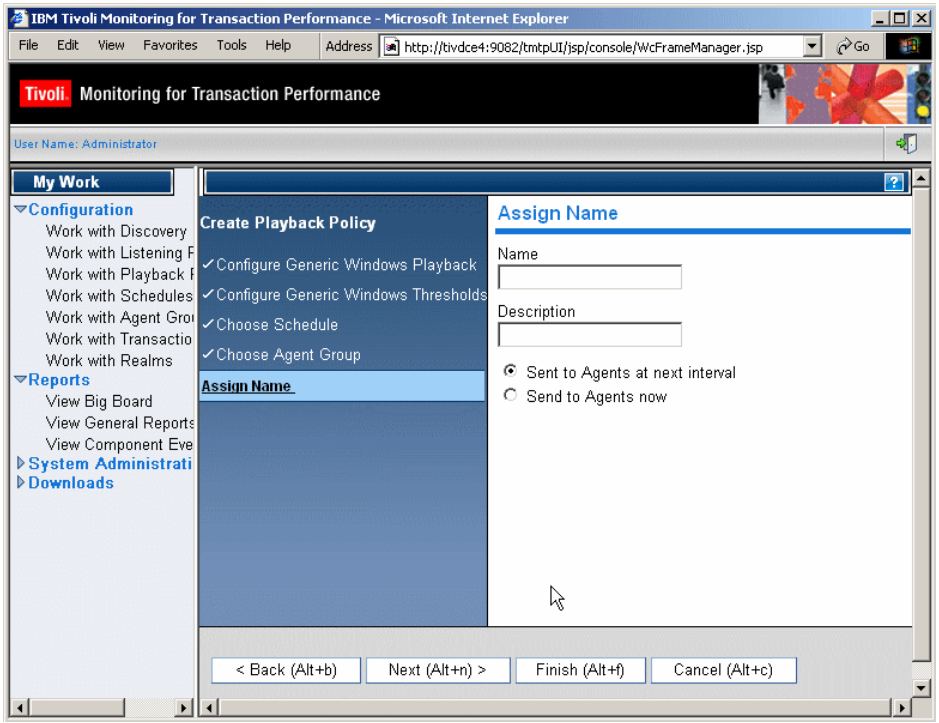


Figure 9-46 Assign your playback policy a name

8. Press the **Finish** button. The Rational Robot scripts associated with your transaction recording will now be pushed out from the Management Server to the Rational Project located on each of the Management Agents in the specified Agent Group, and the associated schedule will be applied to script execution.

10

Historical reporting

This chapter discusses methods and processes of collecting business transaction data from a TMTP Version 5.2 relational database for a Tivoli Enterprise Data Warehouse and performing analysis and presentation of data from a business point of view.

In this chapter, we introduce a new feature of the IBM Tivoli Monitoring for Transaction Performance Version 5.2 warehouse enablement pack (ETL2), and show how to create business reports by using the Tivoli Enterprise Data Warehouse report interface and other OLAP tools.

This chapter provides discussions regarding the following:

- TEDW methods and process
- Configuration and collection of historical data
- Sample e-business transaction and availability report by the TEDW Report Interface
- Customized report by OLAP tools, such as Crystal Enterprise

© Copyright IBM Corp. 2003. All rights reserved.

10.1 TMTP and Tivoli Enterprise Data Warehouse

One of the important features of IBM Tivoli Monitoring for Transaction Performance Version 5.2 is the integration of the common Tivoli repository for historical data, that is, Tivoli Enterprise Data Warehouse. Both the Enterprise- and the Web Transaction Performance features provide these capabilities by supplying functions to extract historical data from the TMTP database.

The Tivoli Enterprise Data Warehouse (TEDW) is used to collect and manage data from various Tivoli and non-Tivoli system management applications. The data is imported into the TEDW databases through specialized extract, transform, and load (ETL) programs, from the management application databases, and further processed for historical analysis and evaluation. It is Tivoli's strategy to provide ETLs for most Tivoli components so the TEDW databases can be populated with meaningful systems management data. IBM Tivoli Monitoring for Transaction Performance is but one of many products to leverage and use TEDW.

10.1.1 Tivoli Enterprise Data Warehouse overview

Having access to historical data regarding the performance and availability of IT resources is very useful in various ways, such as:

- TEDW collects historical data from many applications into one central place.

  TEDW collects the underlying data about the network devices/connections, desktops/servers, applications/software, problems and activities that manage the infrastructure. This allows for the construction of an end-to-end view of the enterprise and viewing of the related resource data independent of the specific applications used to monitor and control the resources.

- TEDW adds value to raw data.

  TEDW performs data aggregation based on user specified periods, such as daily or weekly, and allows for restricting the amount of data stored in the central data TEDW repository. The data is also cleaned and consolidated in order to allow the data model of the central repository to share common dimensions. For example, TEDW ensures that the time, host name, and IP address are the same dimensions across all the applications.

- TEDW allows for correlation of information from many Tivoli applications.

  TEDW can also be used to derive added value by correlating data from many Tivoli applications. It allows reports to be written, which correlate cross application data.

- TEDW uses open, proven interfaces for extracting, storing, and sharing the data.

  TEDW can extract data from any application (Tivoli and non-Tivoli) and store it in a common, central database. TEDW also provides transparent access for third-party Business Intelligence (BI) solutions using the CWM standard, such as IBM DB2 OLAP, Crystal Decisions, Cognos, BusinessObjects, Brio Technology, and Microsoft OLAP Server. CWM stands for Common Warehouse Metadata, an industry standard specification for metadata interchange defined by the Object Management Group (see http://www.omg.org). TEDW provides a Web-based reporting front end called the Reporting Interface, but the open architecture provided by the TEDW allows other BI front ends to be used to access the data in the central warehouse. The value here is *flexibility*. Customers can use the reporting application of their choice; they are not limited to any specific one.

- TEDW provides a robust security mechanism.

  TEDW provides a robust security mechanism by allowing data marts to be built with data from subsets of managed resources; by providing database level authorization to access those data marts, TEDW can address most of the security requirements related to limiting access to specific data to those customers/business units with a need to know.

- TEDW provides a scalable architecture.

  Since TEDW depends on the proven and industry standard RDBMS technology, it provides a scalable architecture for storing and retrieving the data.

Tivoli Enterprise Data Warehouse concepts and components

This section discusses the key concepts and the various components of TEDW in the logical order that the measurement data flows: from the monitors collecting raw data to the final detailed report. Figure 10-1 on page 378 depicts a typical Tivoli Enterprise Data Warehouse configuration that will be used throughout this section

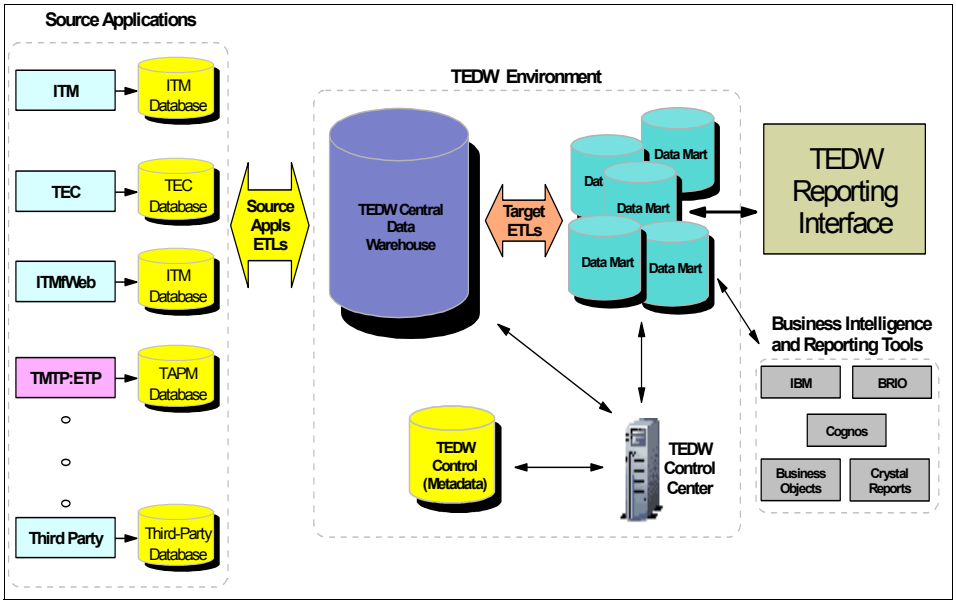


Figure 10-1 A typical TEDW environment

It is common for enterprises to have various distributed performance and availability monitoring applications deployed that collect some sort of measurement data and provide some type of threshold management, central event management, and other basic monitoring functions. These applications are referred as *source applications*.

The first step to obtaining management data is to enable the source applications. This means providing all the tools and castigation necessary to import the source operational data into the TEDW central data warehouse. All components needed for that task are collected in so-called *warehouse modules* for each source application. In this publication, IBM Tivoli Monitoring for Web Infrastructure is the source application providing management data for Web server and Application server data warehouse modules.

One important part of the warehouse modules are the Extract, Transform, and Load data programs, or simply *ETL programs*. In general, ETL programs process data in three steps.

1. First they extract the data from a source application database, called the *data source*.
2. Then the data is validated, transformed, aggregated, and/or cleansed so that it fits the format and needs of the data target.
3. Finally, the data is loaded into the target database.

In TEDW, there are two types of ETLs: *central data warehouse ETL* and *data mart ETL*:

Central data warehouse ETL
: The central data warehouse ETL pulls the data from the source applications and loads it into the central data warehouse, as shown in Figure 10-1 on page 378. The central data warehouse ETL is also often referred to as the *source ETL* or *ETL1*.

Data mart ETL
: As shown in Figure 10-1 on page 378, the data mart ETL extracts a subset of historical data from the central data warehouse that contains data tailored to and optimized for a specific reporting or analysis task. This subset of data is used to populate data marts. The data mart ETL is also known as *target ETL* or *ETL2*.

As a generic concept, a *data warehouse* is a structured, extensible database environment designed for the analysis of consistent data. The data that is inserted in a data warehouse is logically and physically transformed from multiple source applications, updated, and maintained for a long time period of time, and summarized for quick analysis. The Tivoli Enterprise Data Warehouse *Central Data Warehouse (*CDW) is the database that contains all enterprise-wide historical data, with an hour as the lowest granularity. This data store is optimized for the efficient storage of large amounts of data and has a documented format that makes the data accessible to many analysis solutions. The database is organized in a very flexible way, which lets you store data from new applications without adding or changing tables.

The *TEDW server* is an IBM DB2 Universal Database Enterprise Edition server that hosts the TEDW Central Data Warehouse databases. These databases are populated with operational data from Tivoli and/or other third-party applications for historical analyses.

A *data mart* is a subset of the historical data that satisfies the needs of a specific department, team, or customer. A data mart is optimized for interactive reporting and data analysis. The format of a data mart is specific to the reporting or analysis tool you plan to use. Each application that provides a data mart ETL creates its data marts in the appropriate format.

TEDW provides a *Report Interface* (RI) that creates static two-dimensional reports of your data using the data marts. The Report Interface is a role-based Web interface that can be accessed with a simple Web browser without any additional software installed on the client. You can also use other tools to perform OLAP analysis, business intelligence reporting, or data mining.

The *TEDW Control Center* is the IBM DB2 Universal Database Enterprise Edition server containing the TEDW control database that manages your TEDW environment. From the TEDW Control Center, you can also manage all source applications databases in your environment. The default internal name for the TEDW control database is TWH_MD. The TEDW Control Center also manages the communication between the various components, such as the TEDW Central Data Warehouse, the data marts, and the Report Interfaces. The TEDW Control Center uses the DB2 Data Warehouse Center utility to define, maintain, schedule, and monitor the ETL processes.

The TEDW stores raw historical data from all Tivoli and third-party application databases in the TEDW Central Data Warehouse database. The internal name of the TEDW Central Data Warehouse database is TWH_CDW. Once the data has been inserted into the TWH_CDW database, it is available for either the TEDW ETLs to load to the TEDW Data Mart database (the internal name of the TEDW Data Mart database is TWH_MART) or to any other application-specific ETL to process the data and load the application-specific data mart database.

10.1.2 TMTP Version 5.2 Warehouse Enablement Pack overview

IBM Tivoli Monitoring for Transaction Performance Version 5.2 has the ability to display the detailed transaction process information as real-time reports. The data is stored in the TMTP database that runs on either DB2 or Oracle database management products. This database is regarded as the source database for the warehouse pack.

When the TMTP real time reporting data is stored in the source database, the central data warehouse database ETL periodically processes (normally once a day) and extracts data from the source database to the central data warehouse database, TWH_CDW. Once in the central database, the data is converted to the TMTP warehouse pack data model shown in Figure 10-2 on page 381. This data model allows the TMTP reporting data to fit into the general schema of Tivoli Enterprise Data Warehouse Version 1.1.

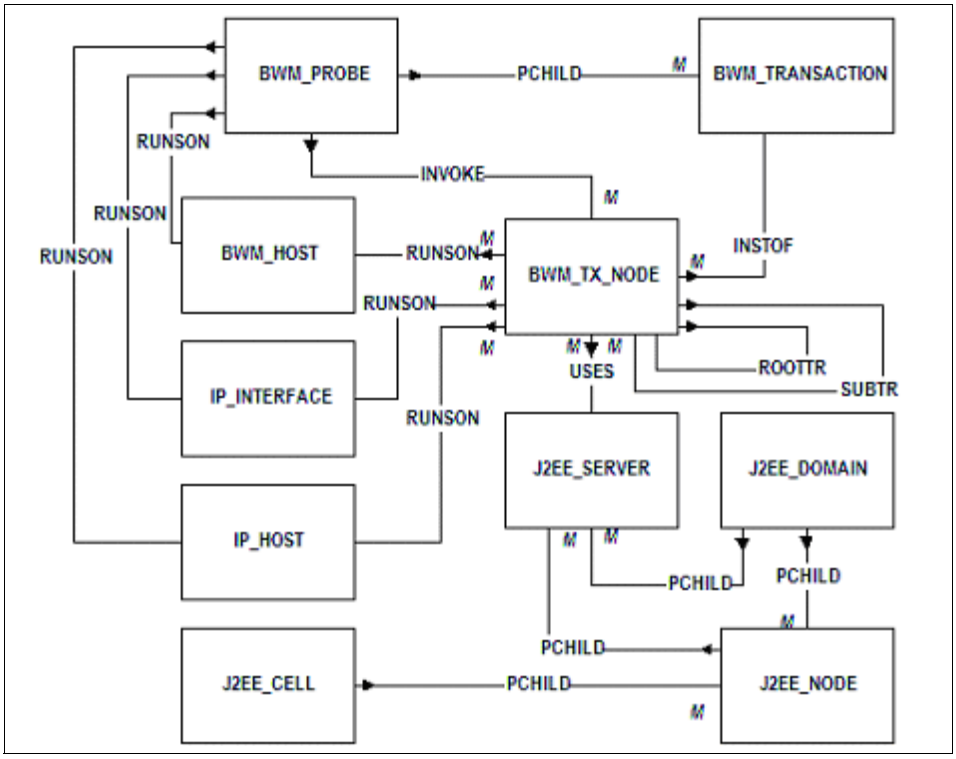


Figure 10-2 TMTP Version 5.2 warehouse data model

After the central data warehouse ETL processes are complete, the data mart ETL processes load data from the central data warehouse database into the data mart database. In the data mart database, fact tables, dimension tables, and helper tables are created in the BWM schema. Data from the central data warehouse database are filled into these dimension and fact tables in the data mart database. You can then use the hourly, daily, weekly, and monthly star schemes of the dimension and fact tables to generate reports in the TEDW report interface.

In addition, the TMTP warehouse pack includes the migration processes for IBM Tivoli Monitoring for Transaction Performance Version 5.1, which enables upgrading existing historical data collected by the IBM Tivoli Monitoring for Transaction Performance Version 5.1 central data warehouse ETL.

IBM Tivoli Monitoring for Transaction Performance does not use resource models; thus, the IBM Tivoli Monitoring warehouse pack and its tables are not required for the TMTP warehouse pack.

10.1.3 The monitoring process data flow

In this section, we will discuss how the warehouse features of both IBM Tivoli Monitoring for Transaction Performance modules interact with the Tivoli Enterprise Data Warehouse. We will also describe the various components that make up the IBM Tivoli Monitoring for Transaction Performance warehouse components. We will demonstrate how the data is collected from the endpoint and how it reaches the data warehouse database, as shown in Figure 10-3. The ETLs used by the warehouse components are explained in Table 10-3 on page 401 and Table 10-4 on page 404.

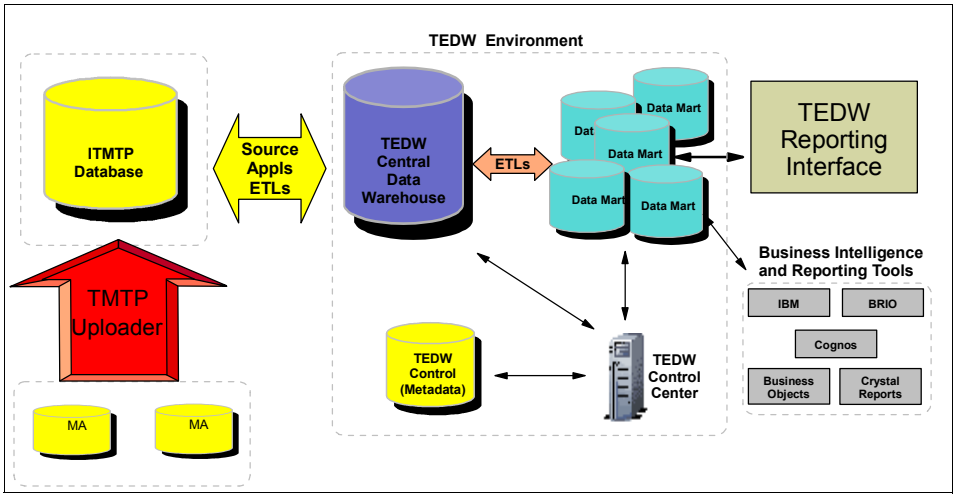


Figure 10-3 ITMTP: Enterprise Transaction Performance data flow

The TMTP upload component is responsible for moving data from the Management Agent to the database. The TMTP ETL1 is then used to collect data from the TMTP database for any module and transform and load these to the staging area tables and dynamic data tables in the central data warehouse (TWH_CDH).

Before going into details of how to install and configure the Tivoli Enterprise Data Warehouse Enablement Packs to extract and store data from the IBM Tivoli Monitoring for Transaction Performance components, the environment used for TEDW in the ITSO lab is presented. This can be used as a starting point for setting up the data gathering process. We assume no preexisting components will be used and describe the steps of a brand new installation.

As shown in Figure 10-4, our Tivoli Enterprise Data Warehouse environment is a small, distributed environment composed of three machines:

1. A Tivoli Enterprise Data Warehouse server machine hosting the central Warehouse and the Warehouse Data Mart databases.
2. A Tivoli Enterprise Data Warehouse Control Center machine hosting the Warehouse meta data database and handling all the ETLs executions.
3. A Tivoli Enterprise Data Warehouse Reporting Interface machine allowing end users to obtain reports from data stored in the data marts.

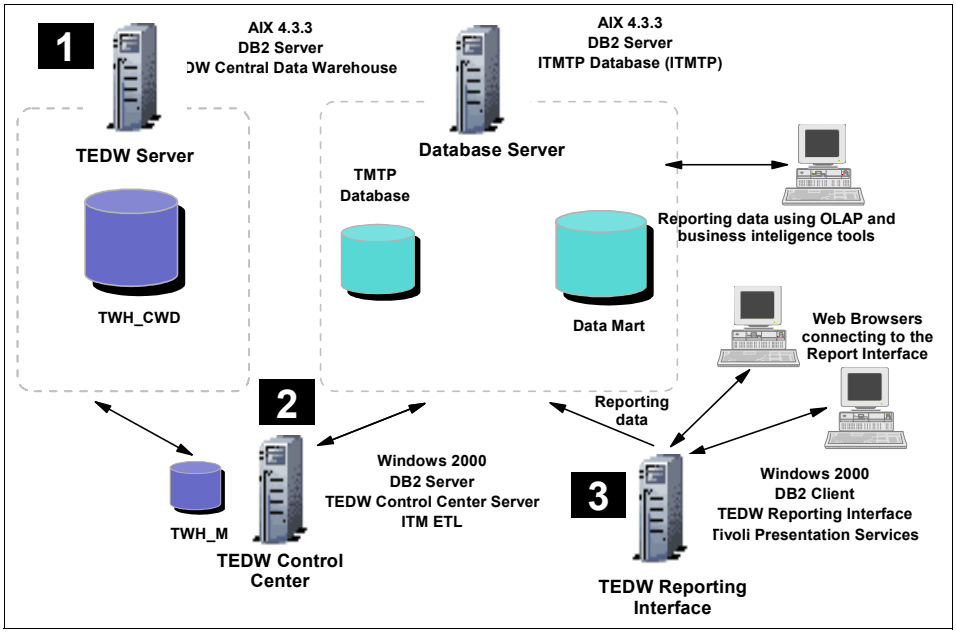


Figure 10-4 Tivoli Enterprise Data Warehouse installation scenario

10.1.4 Setting up the TMTP Warehouse Enablement Packs

The following sections describe the procedures that need to be performed in order to install, configure, and schedule the warehouse modules for the IBM Tivoli Monitoring for Transaction Performance product. The description of the installation steps is based on our lab environment scenario described in Figure 10-4.

It is assumed that the Tivoli Enterprise Data Warehouse environment Version 1.1 is already installed and operational. Details for achieving this can be found in the redbook *Introduction to Tivoli Enterprise Data Warehouse*, SG24-6607.

Throughout the following sections, the Warehouse Enablement Pack for IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance will be used to demonstrate the tasks that needs to be performed, and the changes needed to implement the Warehouse Enablement Pack for IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance will be noted at the end of the walkthrough.

The installation and configuration of the Warehouse Enablement Packs is a four step process that consists of:

Pre-installation steps
: These steps have to be performed to make sure that the TEDW environment is ready to receive the TMTP Warehouse Enablement Packs.

Installation
: The actual transferral of code from the installation images to the TEDW server, and registration of the TMTP ETLs in the TEDW registry.

Post-installation steps
: Provides additional configuration information to ensure the correct function of the TMTP Warehouse Enablement Packs.

Activation
: Includes scheduling and transfer to production mode of the TMTP specific ETL tasks.

Pre-installation steps

Prior to the installation of the Warehouse modules, you must perform the following tasks:

1. Upgrade to DB2 UDB Server Version 7.2 FixPack 6 or higher.
2. Apply TEDW FixPack 1.1-TDW-002 or higher.
3. Update the TEDW environment to FixPack 1-1-TDW-FP01a.
4. Ensure adequate heap size of the TWH_CDW database.

You are only required to perform these steps once, since they apply to the general TWDW environment and not to any specific ETLs.

Upgrade to DB2 UDB Server Version 7.2 FixPack 6 or higher

Upgrade IBM DB2 Universal Database Enterprise Edition Version 7.2 to at least FixPack 6 on your Tivoli Enterprise Data Warehouse environment.

FixPack 6 for IBM DB2 Universal Database Enterprise Edition can be download from the official IBM DB2 technical support Web site:

http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v7fphist.d2w/report

Apply TEDW FixPack 1.1-TDW-002 or higher

Apply the FixPack 1.1-TDW-0002 on every database server in your TEDW environment

FixPack 1.1-TDW-0002 for Tivoli Enterprise Data Warehouse can be downloaded from the IBM Tivoli Software support Web site, under the Tivoli Enterprise Data Warehouse category:

http://www.ibm.com/software/sysmgmt/products/support/

Update the TEDW environment to FixPack 1-1-TDW-FP01a

FixPack 1-1-TDW-FP01a for Tivoli Enterprise Data Warehouse can be downloaded from the IBM Tivoli Software support Web site, under the Tivoli Enterprise Data Warehouse category:

http://www.ibm.com/software/sysmgmt/products/support/

The documentation that accompanies the FixPacks details the steps for installation in greater detail.

Ensure adequate heap size of the TWH_CDW database

The applications control heap size on the TWH_CDW database needs to be set to at least 512 as follows:

1. Log on using the DB2 administrator user ID to your TEDW Server machine (in our case, db2admin), and connect to the TWH_CDW database:

   ```
   db2 connect to TWH_CDW user db2admin using <db2pw>
   ```

   where <db2pw> is the database administrator password.

2. In order to determine the actual heap size issue:

   ```
   db2 get db cfg for TWH_CDW | grep CTL_HEAP
   ```

   The output should be similar to what is shown in Example 10-1.

Example 10-1 Current applications control heap size on the TWH_CDW database

```
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 128
```

3. If the heap size is less that 512, perform:

   ```
   db2 update db cfg for TWH_CDW using APP_CTL_HEAP_SZ 512
   ```

   The output should be similar what is shown in Example 10-2 on page 386.

Example 10-2 Output from db2 update db cfg for TWH_CDW

```
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB21026I  For most configuration parameters, all applications must disconnect
from this database before the changes become effective.
```

4. You should now restart DB2 by issuing the following series of commands:

```
db2 disconnect THW_CDW
db2 force application all
db2 terminate
db2stop
db2admin stop
db2admin start
db2start
```

Limitations

This warehouse pack must be installed using the user db2. If that is not the user name used when installing the Tivoli Enterprise Data Warehouse core application, you must create a temporary user table space for use by the installation program. The temporary user table space that is created in each central data warehouse database and data mart database during the installation of Tivoli Enterprise Data Warehouse is accessible only to the user that performed the installation. If you are installing the warehouse pack using the same database user that installed Tivoli Enterprise Data Warehouse, or if your database user has access to another user temporary table space in the target databases, no additional action is required. If you do not know the user name that was used to install Tivoli Enterprise Data Warehouse, you can determine whether the table space is accessible by attempting to declare a temporary table while connected to each database as the user that will install the warehouse pack. The commands in Example 10-3 are one way to achieve this.

Example 10-3 How to connect TWH_CDW

```
db2 "connect to TWH_CDW user <installing_user> using <password> "
db2 "declare global temporary table t1 (c1 char(1))with replace on commit
preserve rows not logged"
db2 "disconnect TWH_CDW"
db2 "connect to TWH_MART user installing_user using password "
db2 "declare global temporary table t1 (c1 char(1))with replace on commit
preserve rows not logged"
db2 "disconnect TWH_MART"
```

Where:

installing_user Identifies the database user that will install the warehouse pack.

password Specifies the password for the installing user.

Installing the Warehouse Enablement Packs

The IBM Tivoli Monitoring for Transaction Performance Warehouse Enablement Pack extracts data from the ITMTP: Enterprise Transaction Performance RIM database (TAPM*)* and the Web Services Courier database, respectively, and loads it into the TEDW Central Data Warehouse database (TWH_CDW). The two modules acts as a *source* ETLs.

All TEDW ETL programs follow a naming convention using a three letter application-specific product code known as *measurement source code*. Table 10-1 shows the measurement codes used for the TMTP Warehouse Enablement Packs.

Table 10-1 Measurement codes

Warehouse module name	Measurement code
IBM Tivoli Monitoring Transaction and Performance 5.2: WTP	BWM

The installation can be performed using the TEDW Command Line Interface (CLI) or the Graphical User Interface (GUI) based installation program. Here we describe the process using the GUI method.

The following steps should be performed at the Tivoli Enterprise Data Warehouse Control Center server, once for each of the IBM Tivoli Monitoring for Transaction Performance Warehouse Enablement Packs that are being installed.

Note: You need both the TEDW and the appropriate IBM Tivoli Monitoring for Transaction Performance products installation media.

1. Insert the TEDW Installation CD in the CD-ROM drive.
2. Select **Start** → **Run**. Type in `D:\setup.exe` and click **OK** to start the installation, where D is the CD-ROM drive.
3. When the Install Shield Wizard dialogue window for TEDW Installation appears (Figure 10-5 on page 388). Click **Next**.

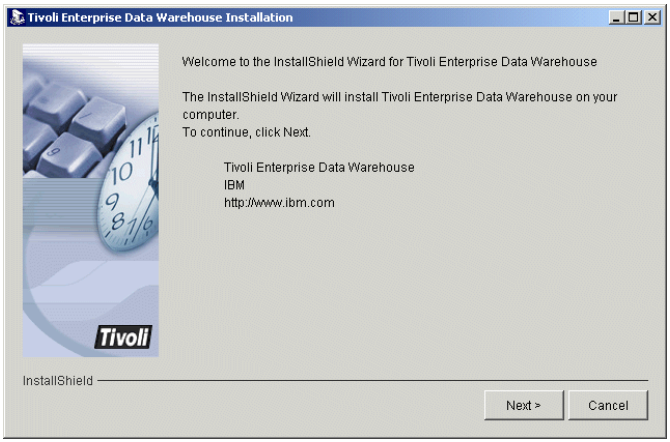


Figure 10-5 TEDW installation

4. The dialog for the type of installation (see Figure 10-6) appears. Select **Application installation only** and the directory name where the TEDW components are installed. We used C:\TWH. Click **Next** to continue.

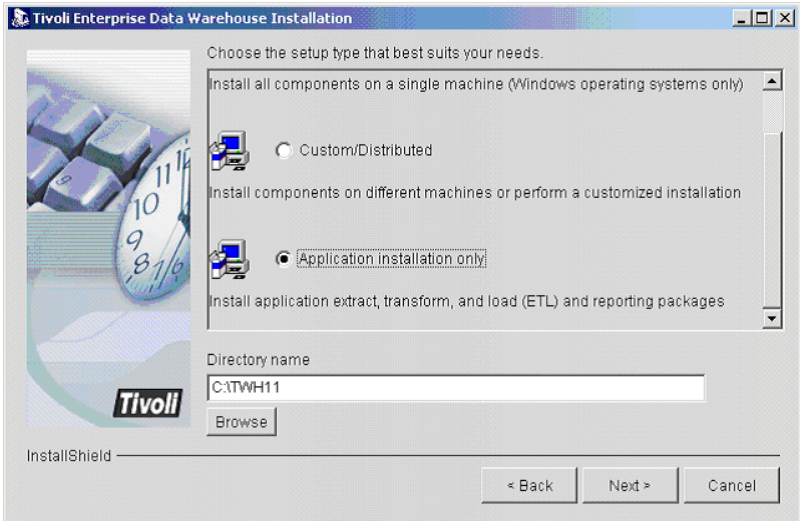

Tivoli Enterprise Data Warehouse Installation
Choose the setup type that best suits your needs.
Install all components on a single machine (Windows operating systems only)
Custom/Distributed
Install components on different machines or perform a customized installation
Application installation only
Install application extract, transform, and load (ETL) and reporting packages
Directory name
C:\TWH11
Browse
Tivoli
InstallShield
< Back
Next >
Cancel

Figure 10-6 TEDW installation type

5. The host name dialog appears, as shown in Figure 10-6. Verify that this is the correct host name for the TEDW Control Center server. Click **Next**
6. The local system DB2 configuration dialog is displayed. It should be similar to what is shown in Figure 10-7 on page 389. The installation process asks for a

valid DB2 user ID. Enter the valid DB2 user ID and password that were created during the DB2 installation on your local system. In our case, we used db2admin. Click **Next**.

Tivoli Enterprise Data Warehouse Installation
Enter the information for configuring DB2 on the local system.
Enter the user name and password to use to access DB2 on the local system.
User name
db2admin
Password
InstallShield
< Back
Next >
Cancel

Figure 10-7 TEDW installation: DB2 configuration

7. The path to the installation media for the application packages dialog appears next, as shown in Figure 10-8.

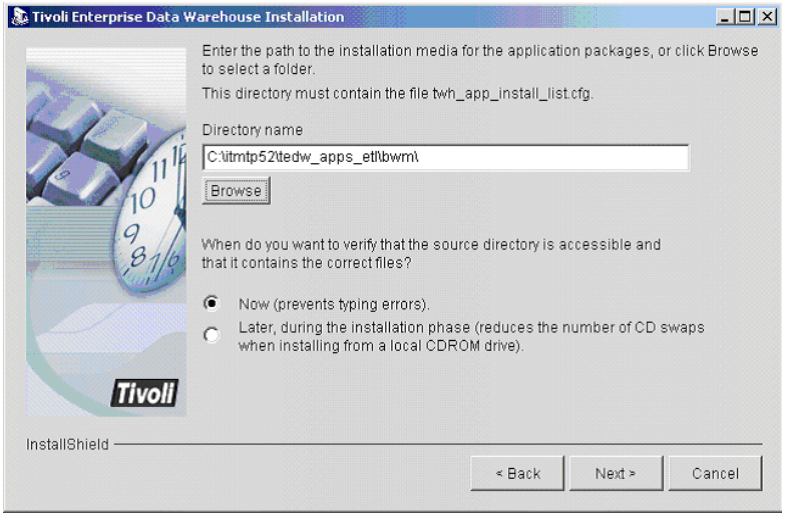

Tivoli Enterprise Data Warehouse Installation
Enter the path to the installation media for the application packages, or click Browse to select a folder.
This directory must contain the file twh_app_install_list.cfg.
Directory name
C:\itmtp52\tedw_apps_etl\bwm\
Browse
When do you want to verify that the source directory is accessible and that it contains the correct files?
Now (prevents typing errors).
Later, during the installation phase (reduces the number of CD swaps when installing from a local CDROM drive).
Tivoli
InstallShield
< Back
Next >
Cancel

Figure 10-8 Path to the installation media for the ITM Generic ETL1 program

You should provide the location of the appropriate IBM Tivoli Monitoring for Transaction Performance ETL1 program. Change the TEDW CD in the CD-ROM drive with the desired installation CD. Specify the path to the installation file named twh_app_install_list.cfg.

If you use the Tivoli product CDs, the paths to the installation files for the ETP and TMTP installation files are:

TMTP <CDROM-drive>:\tedw_apps_etl

Leave the **Now** option checked (prevents typing errors) to verify that the source directory is immediately accessible and that it contains the correct files. Click **Next**.

8. Before starting the installation, do not select to install additional modules when prompted (Figure 10-9). Press **Next**.

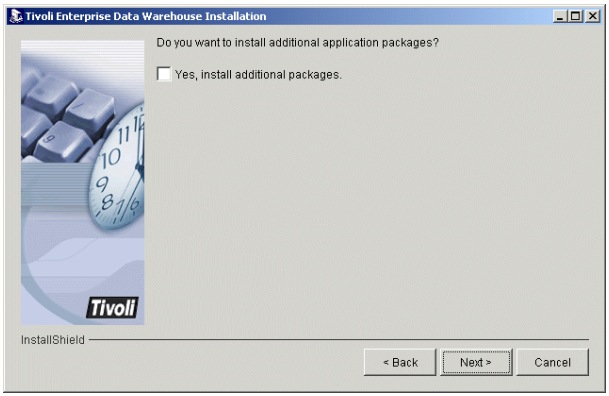


Figure 10-9 TEDW installation: Additional modules

9. The overview of selected features dialogue window appears, as shown in Figure 10-10. Click **Install** to start the installation.

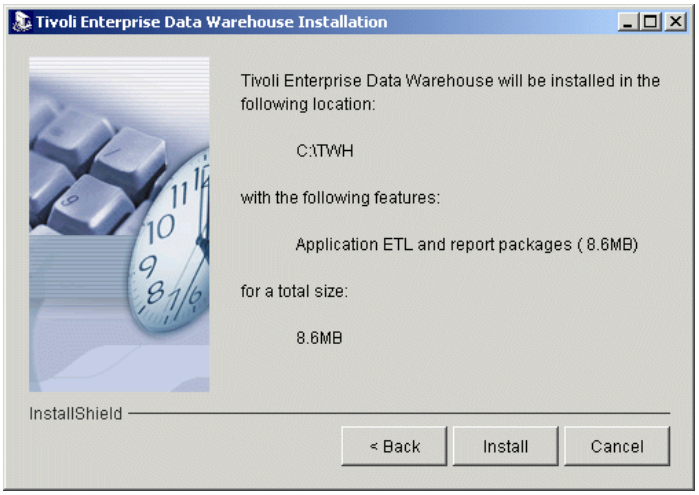


Figure 10-10 TMTP ETL1 and ETL2 program installation

10. During the installation, the panel shown in Figure 10-11, will be displayed. Wait for successful completion.

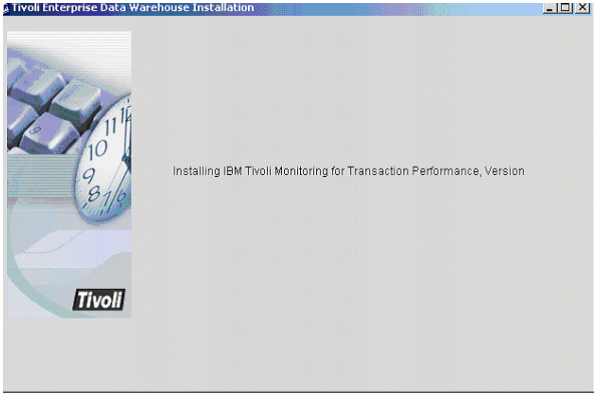

Tivoli Enterprise Data Warehouse Installation
Installing IBM Tivoli Monitoring for Transaction Performance, Version
Tivoli

Figure 10-11 TEDW installation: Installation running

11. Once the installation is finished, the Installation summary dialog appears, as shown in Figure 10-12.

    If the installation was not successful, check the TWHApp.log file for any errors. This log file is located in the *<TWH_inst_dir>*\apps\AMX\, where <TWH_inst_dir> is the TEDW installation directory.

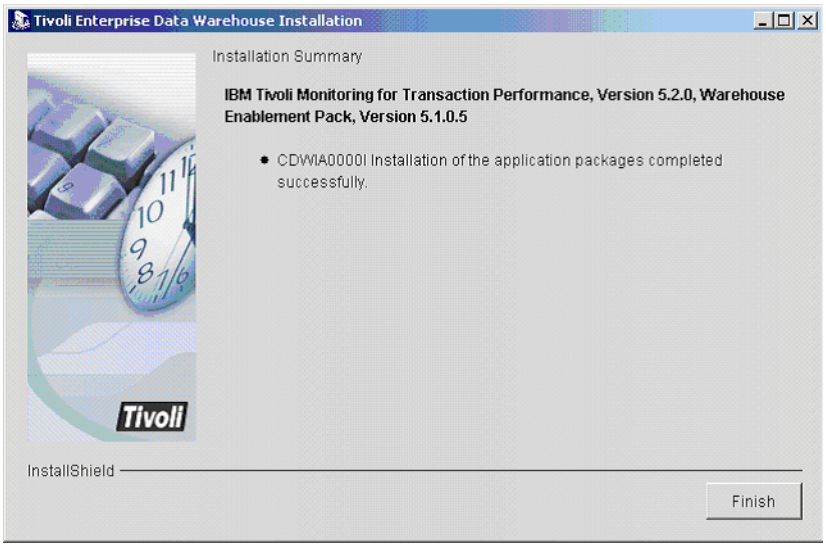

Tivoli Enterprise Data Warehouse Installation
Installation Summary
IBM Tivoli Monitoring for Transaction Performance, Version 5.2.0, Warehouse Enablement Pack, Version 5.1.0.5
CDWIA0000I Installation of the application packages completed successfully.
Tivoli
InstallShield
Finish

Figure 10-12 Installation summary window

Existing TMTP warehouse pack installation

Use the following installation steps to install an existing IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance, Version 5.1.0 warehouse pack Version 1.1.0:

1. Back up the TWH_CDW database before you perform the upgrade.
2. Go to the <TWH_DIR>\install\bin directory.
3. Run the command **sh tedw_wpack_patchadm.sh** to generate a configuration template file. The default file name for the configuration file is <USER_HOME>/LOCALS~1/Temp/twh_app_patcher.cfg. Skip this step if this file already exists.
4. Edit the configuration file to set the parameters to match your installation environment, media location, and user and password settings.
5. Run the **sh tedw_wpack_patchadm.sh** command a second time to install the patch scripts and programs.
6. Open the DB2 Data Warehouse Center.
7. Locate the BWM_c05_Upgrade_Processes group under Subject Areas.
8. Set the schedule for this processes group as execute One Time Only and set the schedule to run immediately. The upgrade process only needs to run once.
9. The upgrade processes defined in this group begin automatically. You can execute the upgrade process without any IBM Tivoli Monitoring for Transaction Performance: Web Transaction Performance Version 5.1.0 historical data. In this case, no data is added into IBM Tivoli Monitoring for Transaction Performance Version 5.2 historical data.

   Set the Version 5.2 central data warehouse ETL and data mart ETL scripts to the Test status to temporarily disable the Version 5.2 central data warehouse ETL processes in the DB2 data warehouse center. This prevents the scripts from automatically executing during the upgrade
10. After the upgrade processes are complete, view the <script_file_name>.log files in the <DB2_HOME>/logging directory to ensure that every script completed successfully.

    A completed message at the end of the log file indicates that the script was successfully performed. If any errors occur, restore the TWH_CDW database from the backup and rerun the processes after problems are located and corrected. A successful upgrade will complete silently and a failed upgrade can stop with or without pop-up error messages in the DB2 data warehouse center. Always check the log files to confirm the upgrade status.
11. Run TMTP data mart ETL processes to extract and load newly upgraded data into the data mart database.

12. Update the user name and password for the Warehouse Sources and Targets in the DB2 Data Warehouse Center.

Note: The BWM_TMTP_DATA_SOURCE must reflect the database where the TMTP Management Server uploads its data. For details on how to update sources and targets, see the *Tivoli Enterprise Data Warehouse Installing and Configuring Guide Version 1.1*, GC32-0744.

Post-installation steps

After successful installation, the following activities must be completed in order to make TEDW suit your particular environment:

1. Creating an ODBC connection to the TMTP source databases
2. Defining user authority to the Warehouse sources and targets
3. Modifying the schema information
4. Customizing your TEDW environment

Creating an ODBC connection to the TMTP source databases

The TEDW Control Center server hosts all the ETLs. This server needs to have access to the various databases accessed by the SQL scripts imbedded in the ETLs. TEDW uses ODBC connections to access all databases, so the TMTP source databases needs to be cataloged at the TEDW DB2 server as ODBC system data sources.

The ETL programs provided with the IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance Warehouse Enablement Packs require specific logical names of the data sources to be used. Table 10-2 shows the values to be used for each of the data sources.

Table 10-2 Source database names used by the TMTP ETLs

Warehouse Enablement Pack	Source database	ETL source database name
TMTP Version 5.2:WTP	TMTP	TMTP_DB_Src

At the TEDW Control Center server, using a DB2 command line window, issue the following commands (in case your source databases are implemented on DB2 RDBMS systems) for each of the source databases:

```
db2 catalog tcpip node <nodename> remote <hostname> server <db2_port>
db2 catalog database <alias> as <database> at node <nodename> ODBC
db2 catalog system odbc data source <alias>
```

Where:

<nodename>	A logical name you assign to the remote DB2 server.
<hostname>	The TCP/IP host name of the remote DB2 server.
<db2_port>	The TCP/IP port used by DB2 (default is 50000).
<alias>	The logical name assigned to the source database. Use the values for the TMTP databases provided in Table 10-2 on page 393.
<database>	The name of the database, as it is known at the DB2 server hosting the database. The values are most likely `TMTP` for Management Server.

Note: If the source databases are implemented using other RDBMS systems (such as Oracle), the commands vary. Instead of using the db2 command line interface, you may use the GUI of the DB2 Client Assistant to catalog the appropriate ODBC data sources. This method may also be used for DB2 hosted source databases.

Defining user authority to the Warehouse sources and targets

You should inform the TEDW Control Center server of user access information for every source and target ETL process installed by the IBM Tivoli Monitoring for Transaction Performance ETL. The following steps should be followed:

1. Start the IBM DB2 Control Center utility by selecting **Start** → **Programs** → **IBM DB2** → **Control Center**.
2. On the IBM DB2 Control Center utility, start the IBM DB2 Data Warehouse Center utility by selecting **Tools** → **Data Warehouse Center**. The Data Warehouse Center logon window appears.
3. Log in to the IBM DB2 Data Warehouse Center utility using the local DB2 administrator user ID, in our case, db2admin.
4. In the Data Warehouse Center window, expand the Warehouse Sources and Warehouse Targets folder. As shown in Figure 10-13 on page 395, there are three entries for the IBM Tivoli Monitoring for Transaction Performance ETL programs that need to be configured:
   - Warehouse Source
     - BWM_TMTP_DATA_SOURCE
     - BWM_TWH_CDW_Source
     - BWH_TWH_MART_Source

– Warehouse Target:
  - BWM_TWH_CDW_Target
  - BWM_TWH_MART_Target
  - BWH_TWH_MD_Target

Edit the properties of each one of the entries above.

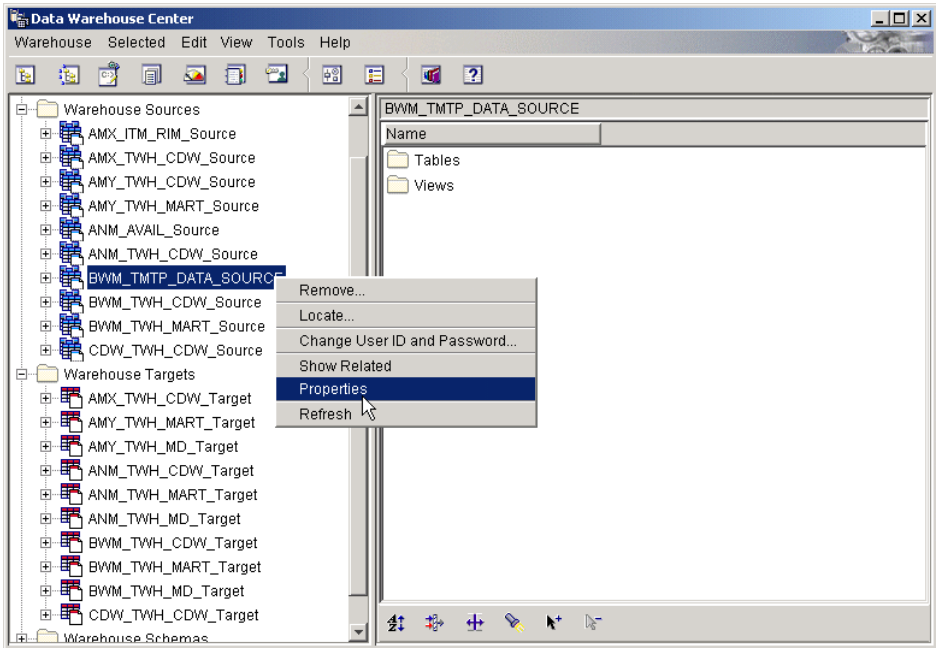


Figure 10-13 TMTP ETL Source and Target

In order to edit the properties of the ETL sources, right-click on the actual object and select **Properties** from the pop-up menu. Then select the **Data Source** tab. Fill in the database instance owner user ID information. For our environment, the values are shown in Figure 10-14 on page 396, using the BWM_TMTP_DATA_SOURCE as an example.

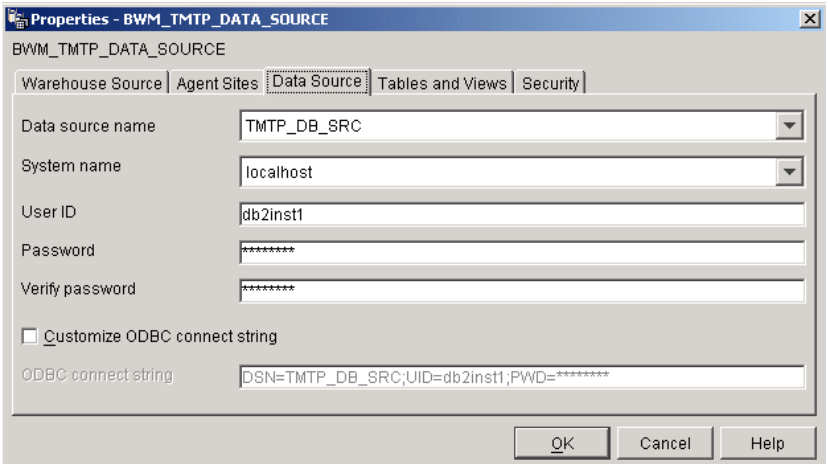


Figure 10-14 BWB_TMTP_DATA_SOURCE user ID information

Set the user ID and password of Data Source for every BWM Warehouse Source and Target ETL.

Modifying the schema information

In order for the ETLs to successfully access the data within the sources defined, an extra step is needed to make sure that the table names referenced by the ETLs match those found in the source databases.

For all the tables used in the IBM Tivoli Monitoring for Transaction Performance Warehouse source (*BWM_TMTP_DATA_SOURCE)* it should be verified that the schema information is filled out, and that the table names do not contain *creator* information. This is, unfortunately, the default situation immediately after installation, as shown in Figure 10-15 on page 397, where you should note that the table names all include the creator information (the part before the period) and the schema field has been left blank.

To provide TEDW with the correct schema and table information, follow the following procedure for every table in each of the IBM Tivoli Monitoring for Transaction Performance ETL sources:

1. On the TEDW Control Center server using Data Warehouse Center window, expand **Warehouse Sources**.
2. Select the appropriate source, for example, BWM_TMTP_DATA_SOURCE, and explode it to be able to see the sub-folders.
3. Open the **Tables** folder.

4. Right-click on each table that appears in the right pane of the Data Warehouse Center window, and select **Properties**. The properties dialog shown in Figure 10-15 appears.

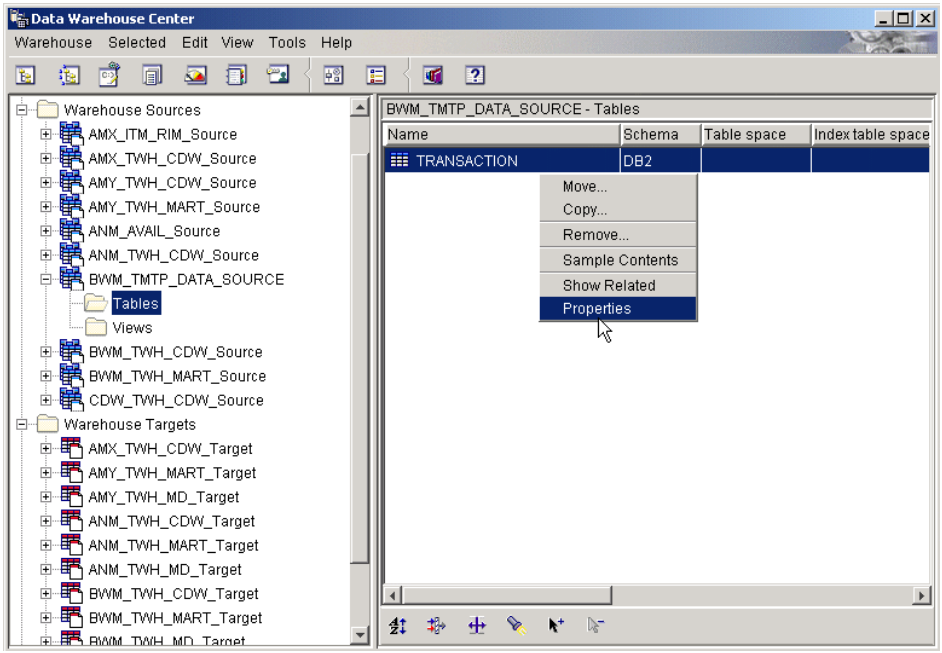


Figure 10-15 Warehouse source table properties

Note that TEDW inserts a default name in the TableSchema field, and that TableName contains the fully qualified name of the table (enclosed in quotes).

5. Type the name of the table creator (or schema) to be used in the TableSchema field, and remove the creator information (including periods and quotes) from the TableName field. The values used in our case are shown in Figure 10-16 on page 398.

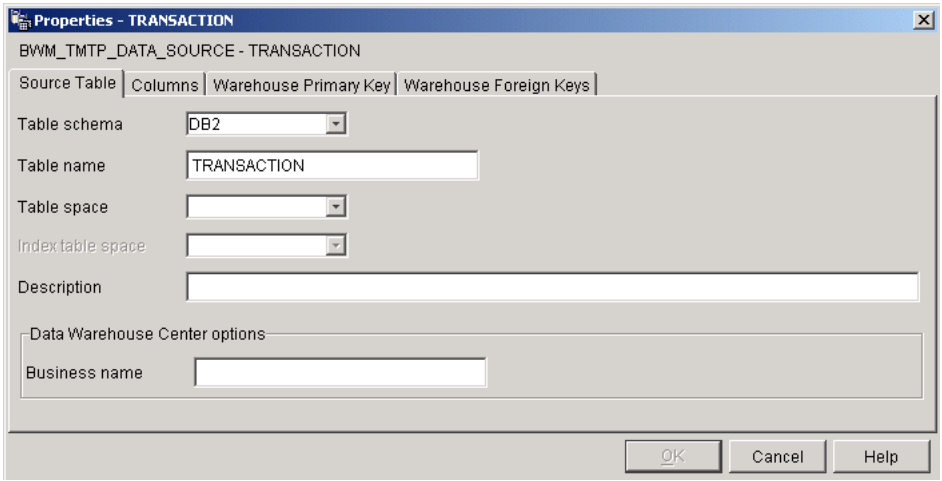


Figure 10-16 TableSchema and TableName for TMTP Warehouse sources

These steps should be performed for all the tables referenced by the two IBM Tivoli Monitoring for Transaction Performance Warehouse sources (*BWM_TMTP_DATA_SOURCE*). Upon completion, the list of tables displayed in the right pane of the Data Warehouse Center window should look similar to the one shown in Figure 10-17, where all the schema information is filled out, and no table names include the creator information.

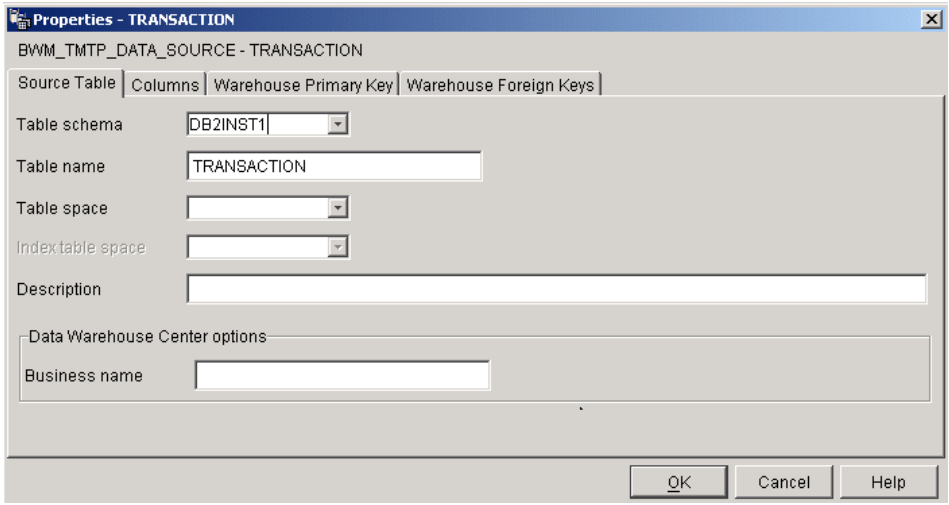


Figure 10-17 Warehouse source table names changed

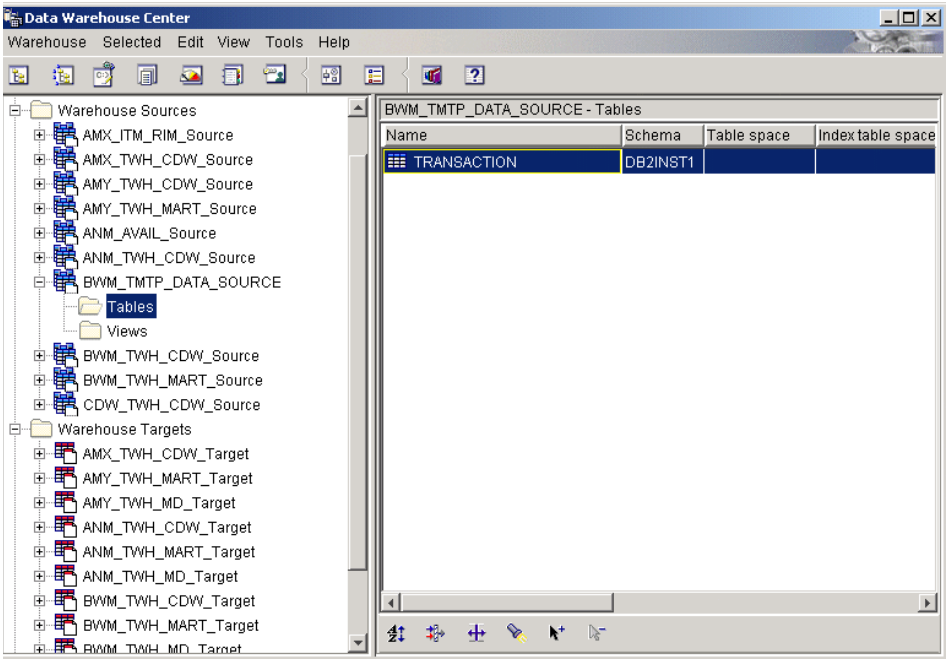


Figure 10-18 Warehouse source table names immediately after installation

Customizing your TEDW environment

After installation of the warehouse enablement pack, use the procedures described in the *Tivoli Enterprise Data Warehouse Installing and Configuring Guide Version 1.1*, GC32-0744 manual to use the Data Warehouse Center to perform the following configuration tasks for data sources and targets:

1. Make sure the control database is set to TWH_MD.
   a. Specify the properties for the BWM_TMTP_DATA_SOURCE data source, ODBC Source.
   b. Set Data Source Name (DSN) to the name of the ODBC connection for the BWM_TMTP_DATA_SOURCE. The default value is DM.
   c. Set the User ID field to the Instance name for the configuration repository. The default value is db2admin.
   d. Set the Password field to the password used to access the BWM_TMTP_DATA_SOURCE.
2. Specify the properties for the target BWM_TWH_CDW_SOURCE.
   a. In the User ID field, type the user ID used to access the Tivoli Enterprise Data Warehouse central data warehouse database. The default value is db2admin.

b. In the Password field, type the password used to access the central data warehouse database.

c. Do not change the value of the Data Source field. It must be TWH_CDW.

3. Specify the following properties for the target BWM_TWH_MART_SOURCE.

a. In the User ID field, type the user ID used to access the data mart database. The default value is db2admin.

b. In the Password field, type the password used to access the data mart database.

c. Do not change the value of the Data Source field. It must be TWH_MART.

4. Specify the properties for the warehouse target BWM_TWH_CDW_TARGET.

a. In the User ID field, type the user ID used to access the central data warehouse database. The default value is db2admin.

b. In the Password field, type the password used to access the central data warehouse database.

c. Do not change the value of the Data Source field. It must be TWH_CDW.

5. Specify the following properties for the target BWM_TWH_MART_TARGET.

a. In the User ID field, type the user ID used to access the data mart database. The default value is db2admin.

b. In the Password field, type the password used to access the data mart database.

c. Do not change the value of the Data Source field. It must be TWH_MART.

6. Specify the properties for the target BWM_TWH_MD_TARGET.

a. In the User ID field, type the user ID used to access the control database. The default value is db2admin.

b. In the Password field, type the password used to access the central data warehouse database.

c. Do not change the value of the Data Source field. It must be TWH_MD.

Specify dependencies between the ETL processes and schedule processes that are to run automatically. The processes for this warehouse pack are located in the BWM_Tivoli_Monitoring_for_Transaction_Performance_v5.2.0 subject area. The processes should be run in the following order:

- BWM_c05_Upgrade51_Process
- BWM_c10_CDW_Process
- BWM_m05_Mart_Process

Attention: Only run the BWM_c05_Upgrade51_Process process if you are migrating from Version 5.1.0 to Version 5.2.

Activating ETLs

Before the newly defined ETLs can start extracting data from the source databases into the TEDW environment, they must be activated. This implies that a schedule must be defined for each of the main processes of the ETLs. After having provided a schedule, it is also necessary to change the operation mode of the all the related ETL components to *production* in order for TEDW to start processing the ETLs according to the specified schedule.

Scheduling the ETL processes

In order to get data extracted periodically from the source database into the data warehouse, a schedule must be specified for all the periodic processes. This is also the case for one-time processes that have to be run to initiate the data warehouse environment for each application area such as TMTP or ITM.

Table 10-3 lists the process that needs to be scheduled for the IBM Tivoli Monitoring for Transaction Performance ETLs to run.

Table 10-3 Warehouse processes

Warehouse enablement pack	Process	Frequency
TMTP:ETL1	BWM_c10_CDW_Process	periodically
TMTP:ETL2	BWM_m05_Mart_Process	periodically

To schedule a process, no matter if it has to run once or multiple times, the same basic steps need to be completed. The only difference between one-time and periodically executed processes is the schedule provided. The following provides a brief walk-trough using the process BWM_c10_CDW_Process to describe the required steps:

1. On the TEDW Control Center server, using the Data Warehouse Center window, expand **Subject Areas**.
2. Select the appropriate Subject Area, for example, BWM_Tivoli_Monitoring_for_Transaction_Performance_v5.2.0_Subject_Area, and explode it to see the processes.
3. Right-click on the process to schedule (in our example, BWM_c10_CDW_Process) and choose **Schedule**, as shown in Figure 10-19 on page 402.

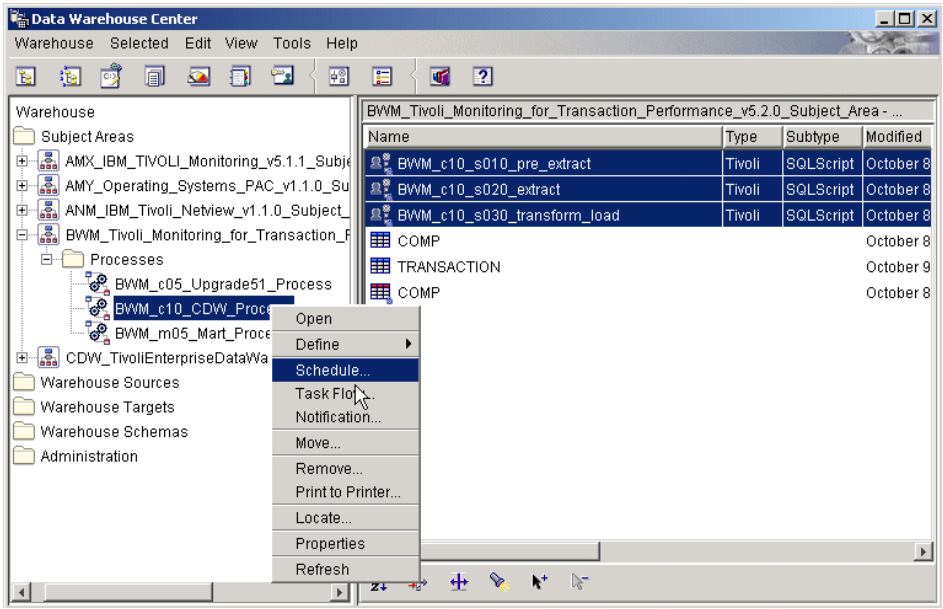


Figure 10-19 Scheduling source ETL process

4. Provide the appropriate scheduling information as it applies to your environment. As shown in Figure 10-20 on page 403, we scheduled the BWM_c10_CDW_Process to run every day at 6 AM.

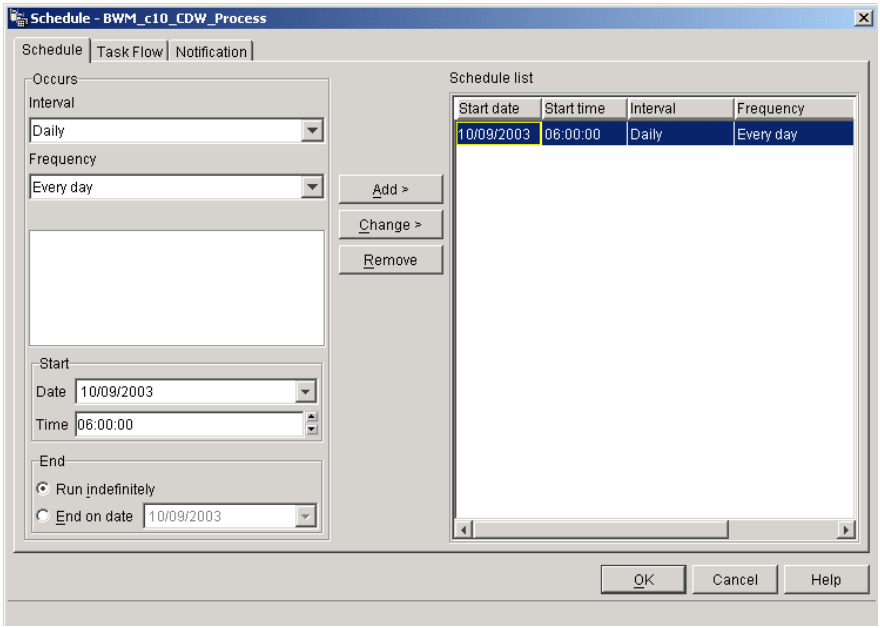


Figure 10-20 Scheduling soure ETL process periodically

Note: To check if the schedule works properly with every process in the source and target ETLS, use the interval setting **One time only**. It may also be used to clear out all previously imported historical information

Figure 10-20 shows an interval of Daily. In general, data import should be scheduled to take place when management activity is low, for example, every night from 2 to 7 AM with a 24 hour interval, or with a very short interval (for example, 15 minutes) to ensure that only small amounts of data have to be processed. The usage pattern (requirements for up-to-date data) of the data in the data warehouse should be used to determine which strategy to follow.

Note: Since TEDW does not allow you to change the schedule once the operation mode has been set to Production, you need to demote the mode of the processes to Development or Test if you want to change the schedule, and do not forget to promote the mode of the processes back to Production to activate the new schedule.

Changing the ETL status to Production

All IBM Tivoli Monitoring for Transaction Performance ETL processes are composed by components that have the Development status set by default. In order for them to run, their status need to be changed from Development to Production.

The following steps must be performed for all processes corresponding to your Warehouse Enablement Pack. Table 10-4 provides the complete list. In the following steps, we use BWM_c10_CDW_Process as an example to describe the process.

Table 10-4 Warehouse processes and components

Warehouse enablement pack	Process	Components
TMTP	BWM_c10_CDW_Process	BWM_c10_s010_pre_extract BWM_c10_s020_extract BWM_c10_s030_transform_load
	BWM_m05_Mart_Proces	BWM_m05_s005_prepare_stage BWM_m05_s010_mart_pre_extrat BWM_m05_s020_mart_extract BWM_m05_s030_mart_load BWM_m05_s040_mart_rollup BWM_m05_s050_mart_prune

On the TEDW Control Center server, using the Data Warehouse Center window, select the desired components and right-click on them. Choose **Mode** → **Production**, as shown in Figure 10-21 on page 405.

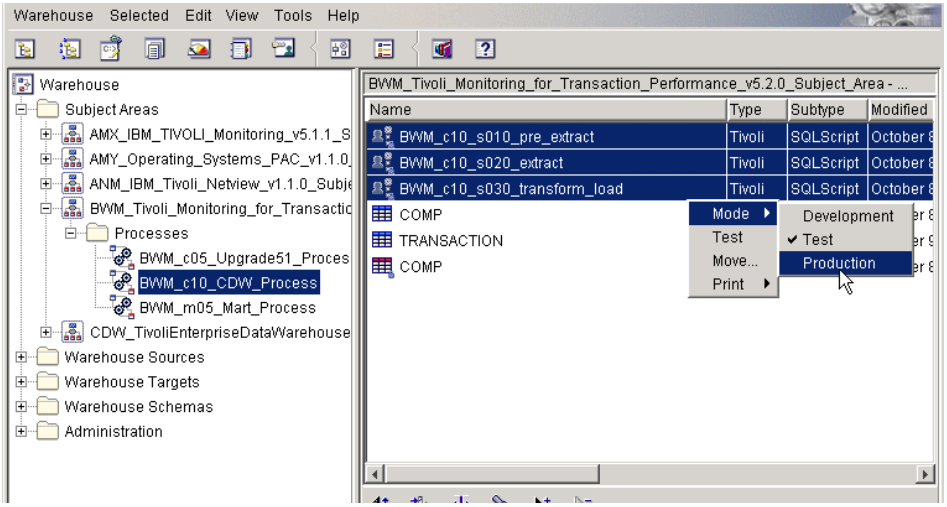


Figure 10-21 Source ETL scheduled processes to Production status

As demonstrated in Figure 10-21, it is possible to select multiple processes and set the desired mode for all of them at the same time.

Now all the process are ready and scheduled to be run in production mode. When the data collection and ETL1 and ETL2 processes are executed, historical data from IBM Tivoli Monitoring for Transaction Performance is available to TMTP Version 5.2 data mart, and you will be ready to generate reports, as described in 10.3.2, “Sample TMTP Version 5.2 reports with data mart” on page 408.

10.2 Creating historical reports directly from TMTP

TMTP Version 5.2 General Reports, such as Overall Transaction Over Time, Availability, and Transaction with Subtransaction, can be used for viewing a transaction report over a short period of time, but are not recommended for reporting over longer periods.

To see a general report of every Trade or Pet Store listening policy and playback policy, navigate to the General Reports, and select the specific type of your interest. Change the settings to view data related to the specific policy and time period of your choice. An example of a Transaction With Subtransaction report is shown in Figure 10-22 on page 406.

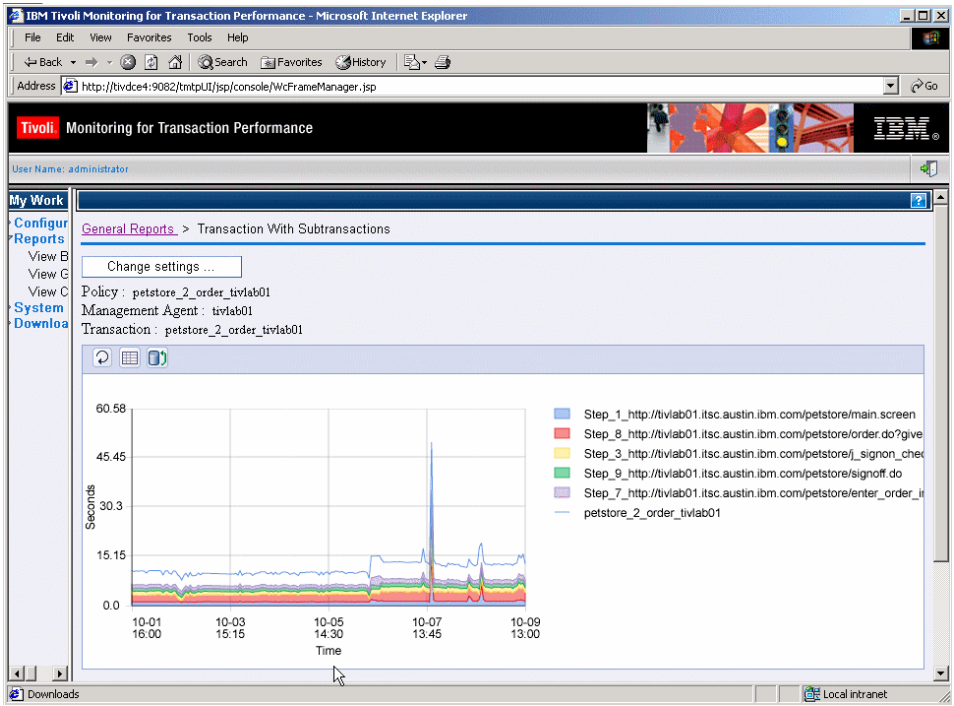


Figure 10-22 Pet Store STI transaction response time report for eight days

Please refer to 8.7, “Transaction performance reporting” on page 295 for more details on using the IBM Tivoli Monitoring for Transaction Performance General Reports.

10.3 Reports by TEDW Report Interface

The following discusses how to use the new TEDW ETL2 reporting feature of IBM Tivoli Monitoring for Transaction Performance Version 5.2.

10.3.1 The TEDW Report Interface

Using the Tivoli Enterprise Data Warehouse Report Interface (RI), you can create and run basic reports against your data marts and publish them on your intranet or the Internet. The Report Interface is not meant to replace OLAP or Business Intelligence tools. If you have multidimensional reporting requirements or need to create a more sophisticated analysis of your data, Tivoli Enterprise Data Warehouse’s open structure provides an easy interface to plug into OLAP or Business Intelligence tools.

Nevertheless, for two-dimensional reporting requirements, Tivoli Enterprise Data Warehouse Report Interface provides a powerful tool. The RI is a role-based Web interface that allows you to create reports from your aggregated enterprise-wide data that is stored in various data marts.

The GUI can be customized for each user. Different roles can be assigned to the users according to the tasks they have to fulfill and the reports they may look at. The users see only those menus in their GUI, which they can use according to their roles. The Report Interface can be accessed with a normal Web browser from everywhere in the network. We recommend using Internet Explorer. Other Web browsers, like Netscape, will also work, but might be slower.

To connect to your Report Interface, start your Web browser and point it to the following URL:

```
http://<your_ri_server>/IBMConsole
```

Where you <your_ri_server> should be replaced by the fully qualified host name of your Report server. The server port is 80 by default. If you chose another port during installation of Tivoli Presentation Services, use the following syntax to start the Report Interface through a different port:

```
http://<your_ri_server>:<your_port>/IBMConsole
```

When you log in for the first time, use the login superadmin and password `password` (you should change this password immediately). After the login, you should see the Welcome page. On the left-hand side, you will find the pane My Work, with all tasks that you may perform.

To manually run a report, complete the following steps:

1. From the portfolio of the IBM Console, select **Work with Reports** → **Manage Reports and Report Output**.
2. In the Manage Reports and Report Output dialog, in the Reports view, right-click on a report icon, and select **Run** from the context menu.

To schedule a report to run automatically when the associated data mart is updated, complete the following steps:

1. From the portfolio of the IBM Console, select **Work with Reports** → **Manage Reports and Report Output**.
2. In the Manage Reports and Report Output dialog, in the Reports view, right-click on a report icon, and select **Properties** from the context menu.
3. Click the **Schedule** option and enable the Run of the report when the data mart is built.

10.3.2 Sample TMTP Version 5.2 reports with data mart

IBM Tivoli Monitoring for Transaction Performance Version 5.2 provides the BWM Transaction Performance data mart. This data mart uses the following star schemas:

- BWM_Hourly_Tranaction_Node_Star_Schema
- BWM_Daily_Tranaction_Node_Star_Schema
- BWM_Weekly_Tranaction_Node_Star_Schema
- BWM_Monthly_Tranaction_Node_Star_Schema

The data mart provides the following pre-packaged health check reports:

- Response time by application
- Response time by host name
- Execution load by application
- Execution load by user
- Transaction availability

all of which are explained in greater detail in the following sections.

Response time by application

This report shows response times during the day for individual applications. Application response time is the average of the response times for all transactions defined within that application. The response time measurement unit is in seconds.

This report uses the BWM_Daily_Transaction_Node_Star_Schema.

The categories shown in the Response Time by Application report in Figure 10-23 on page 409 are labeled J2EE Vendor/ J2EE Version; J2EE Server Name; Probe name. The actual values for this report are:

1. N/A; N/A; STI
2. N/A; N/A; GenWin
3. N/AN/A; .*; N/A
4. WebSphere5.0; server1; N/A
5. N/A; N/A; QOS

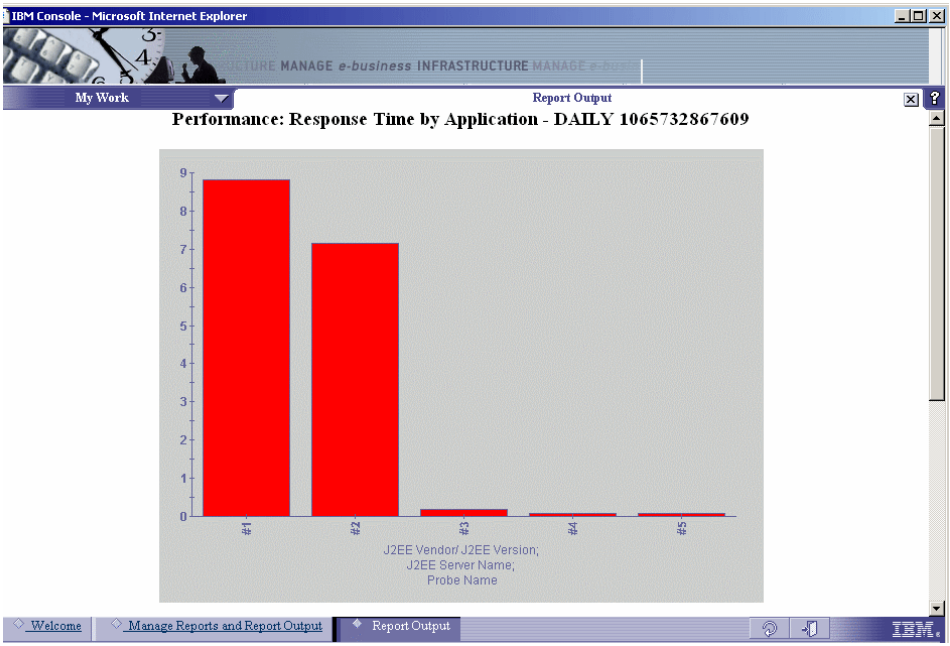


Figure 10-23 Response time by Application

Response time by host name

This report shows response times during the day for individual IP hosts. The complete host name appears as *hostname.domain*. Each host can be a single user machine or a multi-user server. The response time measurement unit is seconds.

The report is based on the BWM_Daily_Transaction_Node_Star_Schema.

The categories shown in the Response Time by Hostname report in Figure 10-24 on page 410 are labeled Transaction Host Name; Probe Host Name. The actual values for this report are:

1. tmtpma-xp.itsc.austin.ibm.com; tmtpma-xp.itsc.austin.ibm.com
2. tivlab01; tivlab01
3. tivlab01; N/A
4. ibmtiv9; ibmtiv9
5. ibmtiv9; N/A

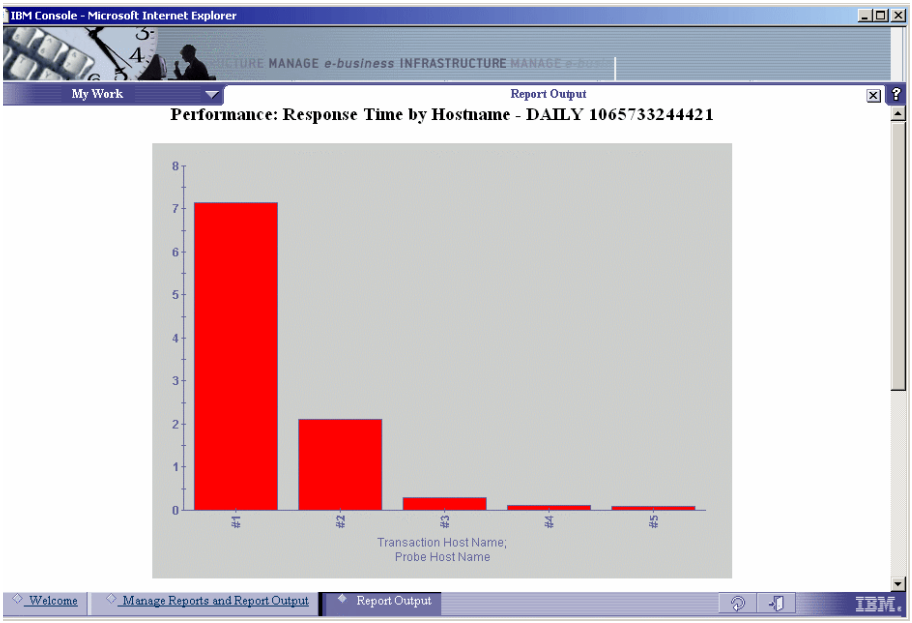


Figure 10-24 Response time by host name

Execution load by application

This report shows the number of times any transaction within the application was run during the time interval. This shows which applications are being used the most. If an application has an unusually low value, it may have been unavailable during the interval.

This report uses the BWM_Daily_Transaction_Node_Star_Schema.

The categories shown in the Execution Load by Application report in Figure 10-25 on page 411 are labeled J2EE Vendor/ J2EE Version; J2EE Server Name; Probe name. The actual values for this report are:

1. WebSphere5.0; server1; N/A
2. N/A; N/A; QOS
3. N/A; N/A; STI
4. N/A; N/A;N/A
5. N/A; N/A;N/A

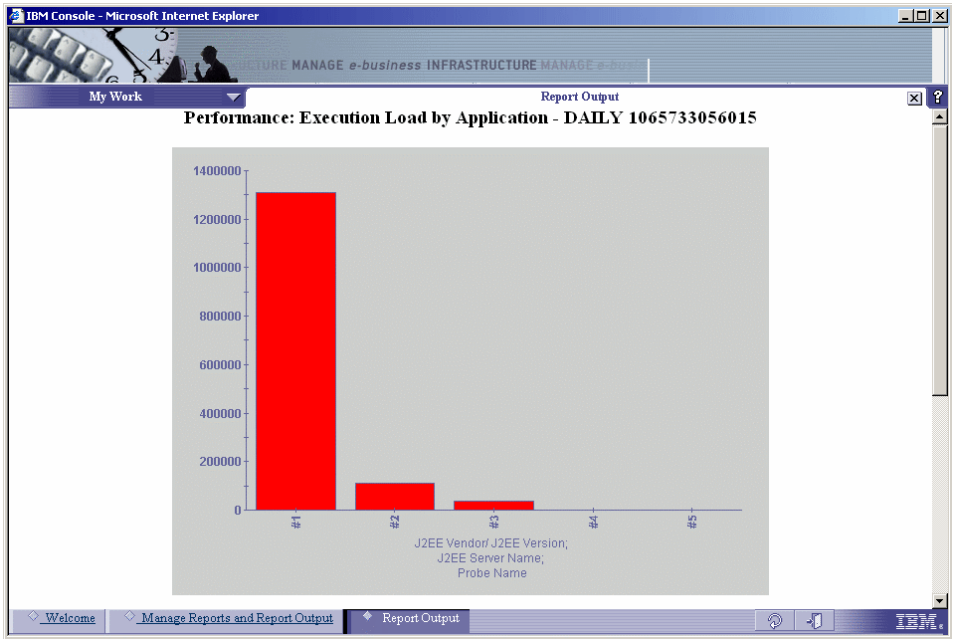


Figure 10-25 Execution Load by Application daily

Execution load by user

This report (Figure 10-26 on page 412) shows the number of times a user has run an application or transaction during the time interval. This shows which users are using the applications and how often they are using them. Such information can be used to charge for application usage. The users names are their user IDs to the operating system. If more than one user logs on with the same user ID, the user ID displayed in the graph may represent more than one user.

This report uses the BWM_Daily_Transaction_Node_Star_Schema.

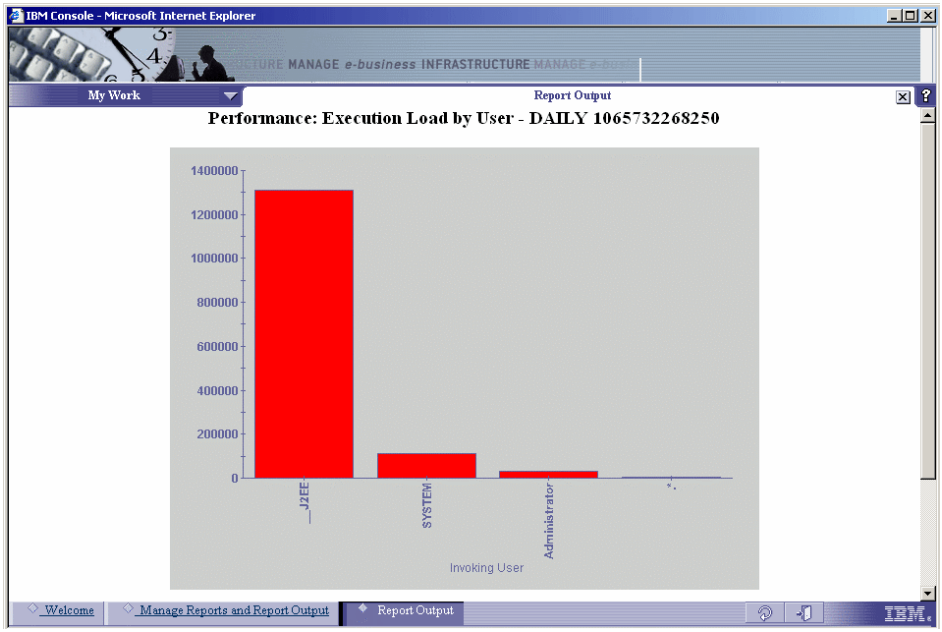


Figure 10-26 Performance Execution load by User

Transaction availability

This report (Figure 10-27 on page 413) shows the availability of a transaction over time in bar chart form. This report uses the BWM_Daily_Transaction_Node_Star_Schema.

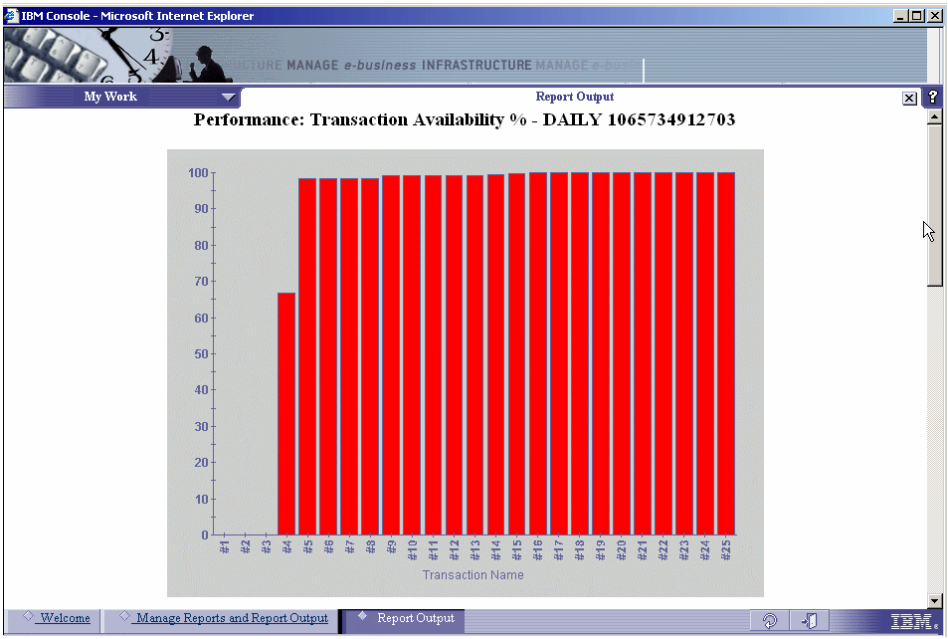


Figure 10-27 Performance Transaction availability% Daily

10.3.3 Create extreme case weekly and monthly reports

The extreme case report type provided by TEDW is a one measurement versus many components type of report. With this type of report, you can find the components or component groups with the highest or lowest values of a certain metric. The result will be a graph with the worst or best components in the x-axis and the corresponding metric values in the y-axis.

In the following sections, we will demonstrate the procedure to create a Weekly Execution Load by User report:

1. Open your IBM Console, expand **Work with Reports** and click **Create Report**.
2. Choose **Extreme Case** from the type selection and proceed. The first difference in the summary report is in the **Add Metrics** dialog. In an extreme case report, you can choose one metric only, for example, the metric with the extreme value. There is one additional field compared to the summary report below the metric list. Here you can change the order direction. If you choose *ascending* order, the graph will start with the lowest value of the metrics. Conversely, you can use *descending* order to find the largest values. As you already chose the order of the graph in this dialog, the Order By choice will be

missing in the Specify Attributes dialog. Select the data mart BWM_Transaction_Performance_Data_Mart and click **OK**.

3. We chose the host name as the Group By entry and the relevant subdomain in the Filter By entry.
4. Check the **Public** button when you want to create a public report that can be seen and used by other users. You see the public entry only when you have sufficient roles to create public reports.
5. Click on the **Metrics** tab. You will see the list of chosen metrics, which is still empty. In a summary report, there a typically many metrics.
6. Click **Add** to choose metrics from the star schema. You will see the list of all star schemes of the chosen data mart (Figure 10-28 on page 415).
7. Select one of them, and you will see all available metrics of this star schema. You see that there is a minimum, maximum, and average type of each metrics. These values are generated when the aggregation of the source data to hourly and daily data is done. Each *aggregation level* has its own star schema with its own fact table. In a fact table, each measurement can have a minimum, maximum, average, and total value. Which values are used depends on the application and can be defined in the D_METRIC table. When a value is used, a corresponding entry will appear in the available metrics list in the Reporting Interface.
8. Choose the metrics you need in your report and click **Next**. You will see the Specify Aggregations dialog. In this dialog, you have to choose an aggregation type for each chosen metric. A summary report covers a certain time window (defined later in this section). All measurements are aggregated over that time window. The aggregation type is defined here.

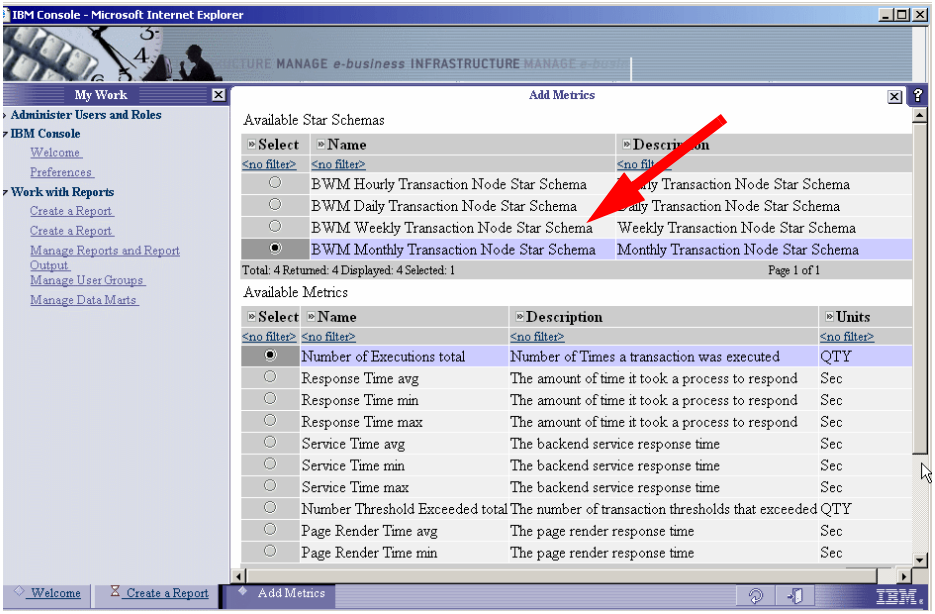


Figure 10-28 Add metrics window

9. With Filter By, you select only those records that match the values given in this file. In the resulting SQL statement, each chosen filter will result in a where clause.

   The Group By function works as follows: if you choose one attribute in the Group By field, then all records with the same value for this attribute are taken together and aggregated according to the type chosen in the previous dialog. The result is one aggregated measurement for each distinct value of the chosen attribute. Each entry in the Group By column will result in a group by clause in the SQL statement. The aggregation type will show up in the select part (line 1) where Total is translated to sum.

10. We chose no filter in our example. The possible choices of the filters are automatically populated from all values in the star schemas. If more than 27 distinct values exist you cannot filter on these attributes (see Figure 10-29 on page 416).

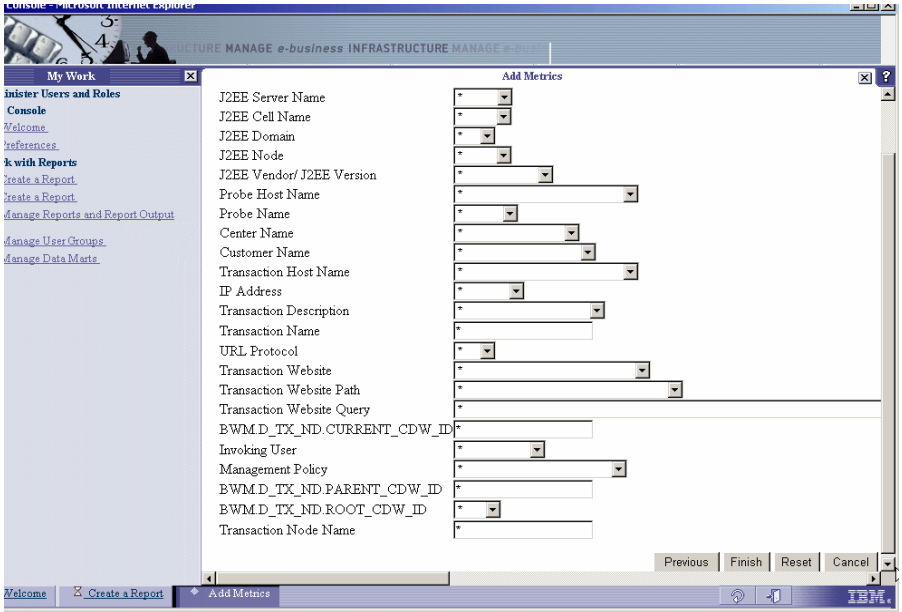


Figure 10-29 Add Filter windows

11. Click **Finish** to set up your metrics and click on the Time pad.
12. In the Time dialog, you have to choose the time interval for the report. In summary reports, all measurements of the chosen time interval will be aggregated for all groups.
13. In the Schedule pad, you can select the **Run** button to execute the report when the data mart is built. A record inserted into the RPI.SSUpdated table in the TWH_MD database tells the report execution engine when a star schema has been updated, and the report execution engine runs all scheduled reports that have been created from that star schema.
14. When all settings are done, click **OK** to create the report. You should see a message window displaying `Report created successfully`.
15. To see the report in the report list, click **Refresh** and expand root in the Reports panel, and click **Reports**, as demonstrated in Figure 10-30 on page 417.

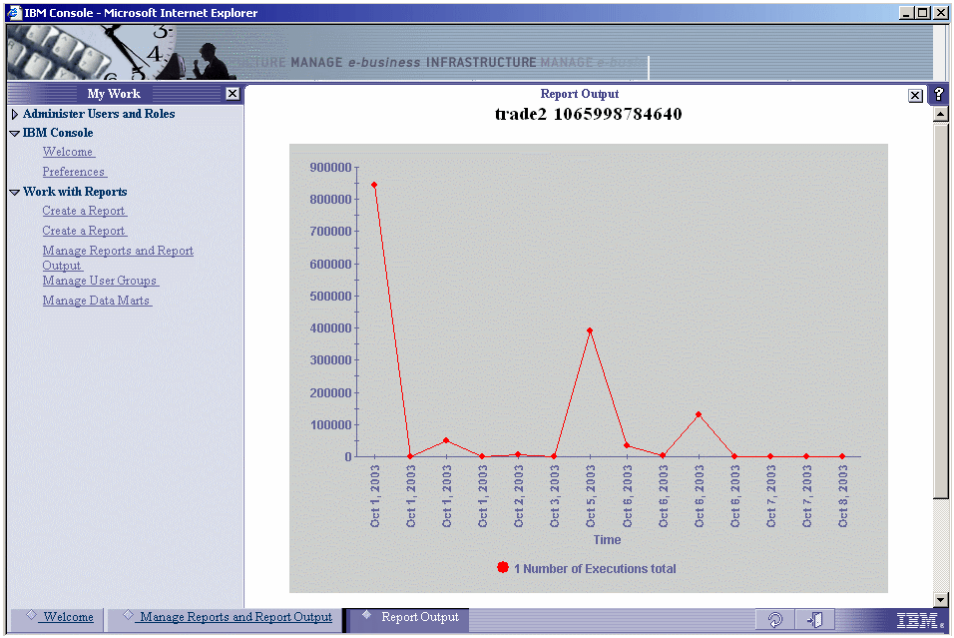


Figure 10-30 Weekly performance load execution by user for trade application

Usually the reports are scheduled and run automatically when the data mart is built. However, you can run the report manually at any time by choosing **Run** from the reports pop-up menu.

You can now save this report output. You will find it in the folder Report Output.

10.4 Using OLAP tools for customized reports

Online Analytical Processing (OLAP) is a technology used in creating decision support software that allows application users to quickly analyze information that has been summarized into multidimensional views and hierarchies. By summarizing predicted queries into multidimensional views prior to run time, OLAP tools can provide the benefit of increased performance over traditional database access tools. OLAP functionality is characterized by dynamic multi-dimensional analysis of consolidated enterprise data supporting end user analytical and navigational activities, including:

- Calculations and modeling applied across dimensions, through hierarchies, and/or across members
- Trend analysis over sequential time periods

- Slicing subsets for on-screen viewing
- Drill-down to deeper levels of consolidation
- Reach-through to underlying detail data
- Rotation to new dimensional comparisons in the viewing area

10.4.1 Crystal Reports overview

The OLAP tools used to demonstrate creation of OLAP reports in the following provides connectivity to virtually any enterprise data source, rich features for building business logic, comprehensive formatting and layout, and high-fidelity output for the Web or print.

In addition, Crystal Reports provides an extensible formula language for building complex reports requiring complex business logic. Built-in interactivity, personalization, parameters, drill-down, and indexing technologies enable custom content to be delivered to any user, based on security or on user-defined criteria. Finally, any report design can be outputted to a variety of formats, including PDF, Excel, Word, and our standard, published XML schema (XML can also be tailored for to match other standard schema).

The value of a standard tool extends beyond the widespread availability and general quality of the product. It includes all the value-add often associated with industry standards: large pools of skilled resources, large knowledge base, partnerships and integration with other enterprise software vendors, easy access to consulting and training, third-party books and documentation, and so on. Standard tools tend to travel with a whole caravan of support and services that help organizations succeed.

Crystal Reports is designed to produce accurate, high resolution output to both DHTML and PDF for Web viewing and printing. Output to RTF enables integration of structured content into Management Server Word documents. Built-in XML support and a standard Report XML schema deliver output for other devices and business processes and native Excel output enables further desktop analysis of report results.

For more information about Crystal Reports go to:

http://www.crystaldecisions.com/

10.4.2 Crystal Reports integration with TEDW

The following section provides information on how to customize and use Crystal Reports to generate OLAP reports based on the historical data in the TEDW database gathered by IBM Tivoli Monitoring for Transaction Performance.

Setting up integration

Follow the following steps to configure Crystal Reports:

1. Install Crystal Reports at your desktop.
2. Install DB2 client at your desktop if you have not installed it already.
3. Create ODBC data sources at your desktop to connect TWH_CDW database.

Crystal Reports and TMTP Version 5.2 sample reports

TWH_CDW database (ETL1) source data has been used here to create TMTP reports through Crystal Reports.

Steps to create a report

Follow the following steps to create a TMTP Version 5.2 report from Crystal Reports:

1. Select **Programs** → **Crystal Reports** → **Using the Report Expert** → **OK** → **Choose an Expert** → **Standard**.
2. Select **Database** → **Open ODBC**.

   Choose the data source that you have created to connect to TWH_CDW database and select the appropriate database ID and password.
3. Choose the COMP, MSMT, and MSMTTYP tables from the TWH_CDW database, as shown in Figure 10-31. Click **Add** and **Next** to create the links.

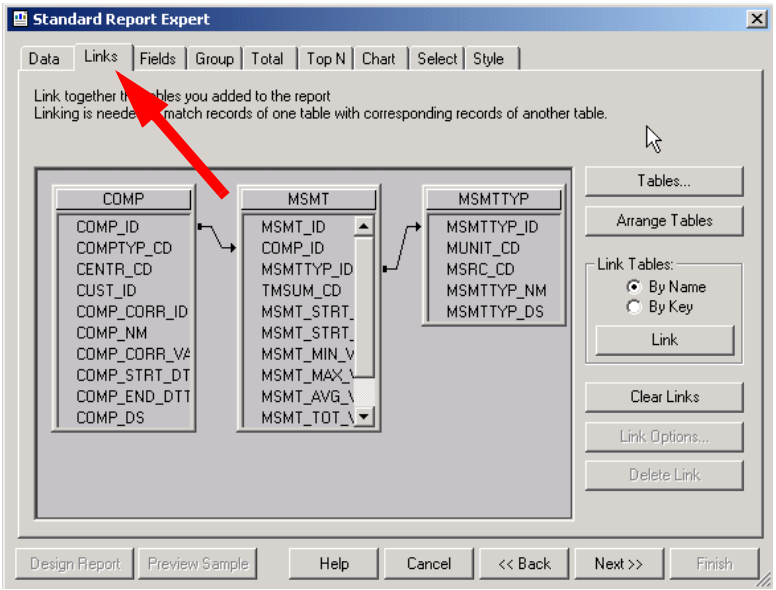


Figure 10-31 Create links for report generation in Crystal Reports

4. Click **Field** and choose fields from the list shown in Figure 10-32.

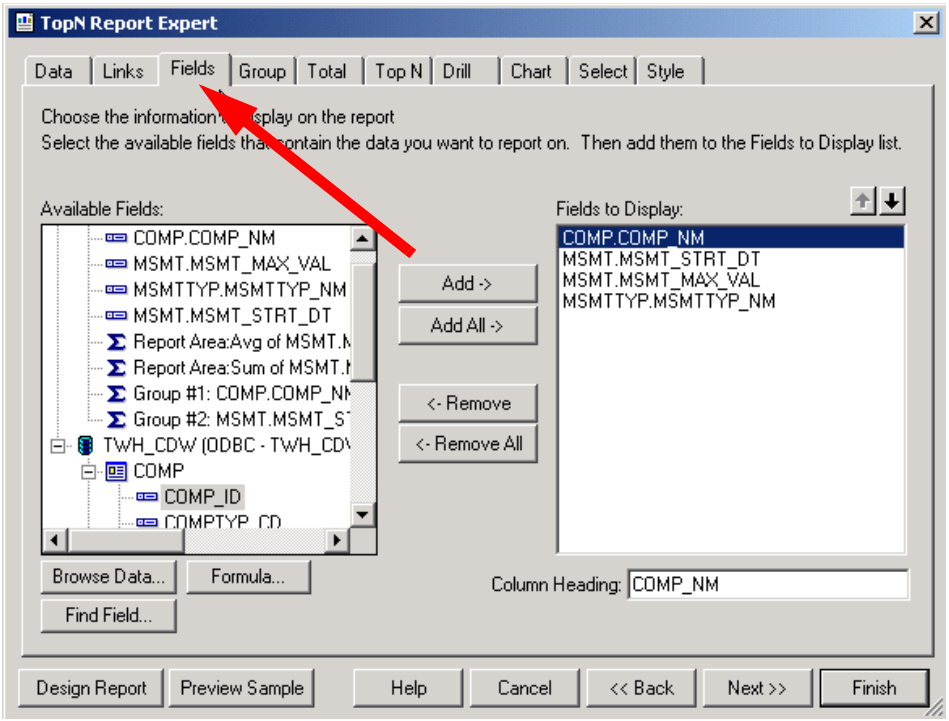


Figure 10-32 Choose fields for report generation

5. Click **Group** and choose the groups COMP.COMP_NM and MSMT.MSMT_STRT_DT.
6. Click **Total** and choose MSMT.MSMT_AVG_VA and summary type *average*.
7. Click **Select** and choose MSMTTYP.MSMTTYP_MN and COMP.COMP_NM to define filtering for your report, as demonstrated in Figure 10-33 on page 421.

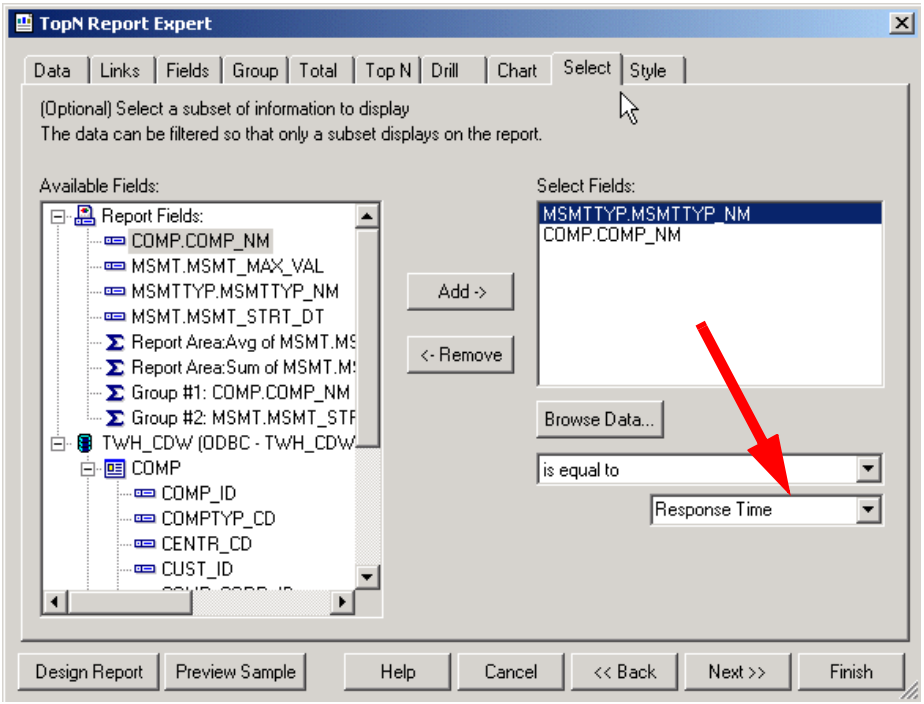


Figure 10-33 Crystal Reports filtering definition

- Provide a title for the report, for example, Telia Trade Stock Check Report, and click **Finish**.

Tip: You can make a filter for the MSMTTYP_NM field and choose different values, such as Response time, Round trip time, Overall Time, and more to create different type of reports.

10.4.3 Sample Trade application reports

We have created the following sample reports using a very simple Crystal Reports report design:

- Average Simulated Response Time by date
- J2EE Response Time by date
- JDBC Response Time by date
- Average End-user Experience by date

You can download the Crystal Reports files containing the report specifications. Please refer to Appendix C, “Additional material” on page 473 for details on how to obtain a copy.

Average Simulated Response Time by date

The Average Simulated Response Time by date report in Figure 10-34 shows that the response times reported by the trade_2_stock-check_tivlab01 played back STI transaction are fairly consistent in the seven second range.

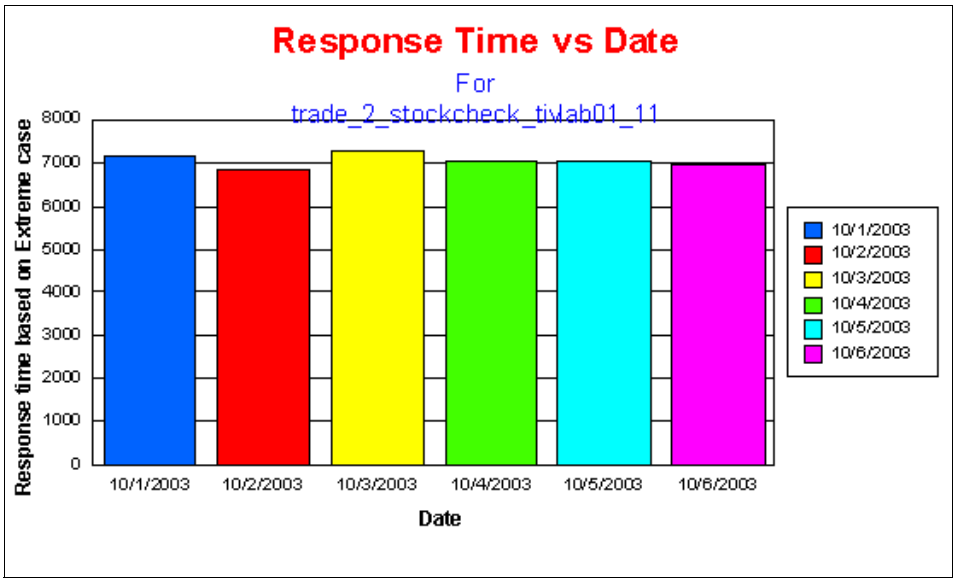


Figure 10-34 trade_2_stock-check_tivlab01 playback policy end-user experience

J2EE Response Time by date

The J2EE Response Time by date report in Figure 10-35 on page 423 shows that special attention needs to be devoted to tuning the J2EE environment to make the response times from the J2EE backed transactions monitored by the trade_j2ee_lis listening policy more consistent.

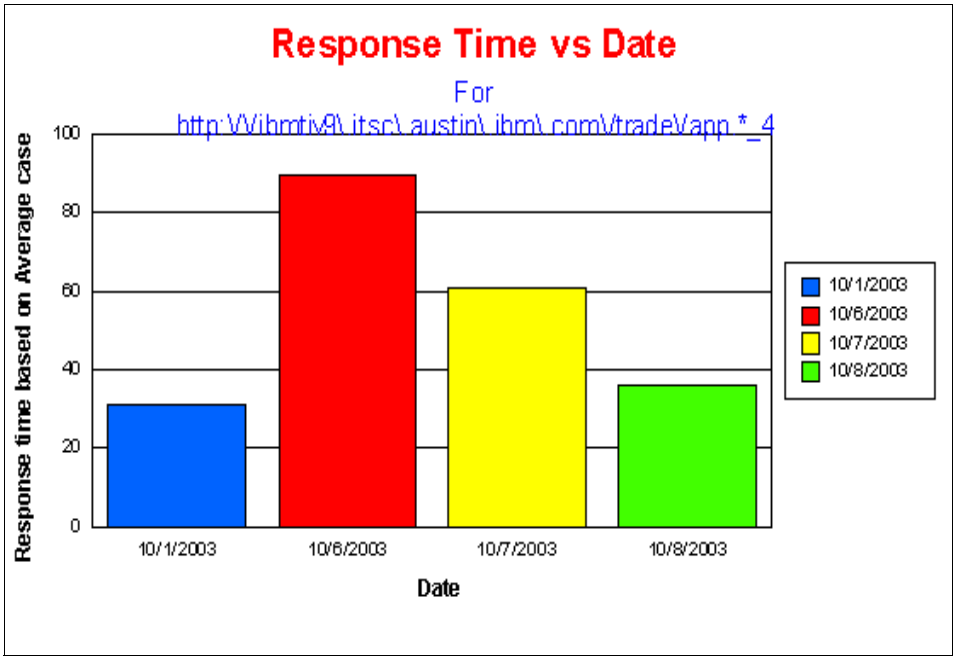


Figure 10-35 trade_j2ee_lis listening policy response time report

The JDBC Response Time by date shown in Figure 10-36 on page 424 shows that after a problematic start on 10/1, the tuning activities of the database has had the desired effect.

JDBC Response Time by date

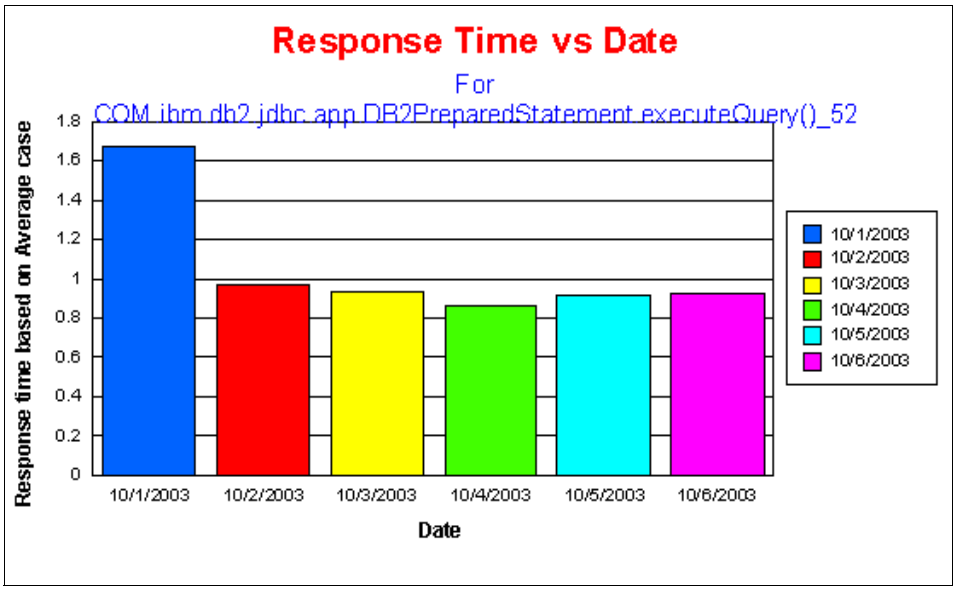


Figure 10-36 Response time JDBC process: Trade applications executeQuery()

Average End-user Experience by date

The Average End-user Experience by date report shown in Figure 10-37 on page 425 reveals that there might be networking issues that are related to the end users running trade transactions on 10/6. This report does not detail the difference in the locations of the active user population between the two days, but it is obvious that troubleshooting and/or tuning is needed.

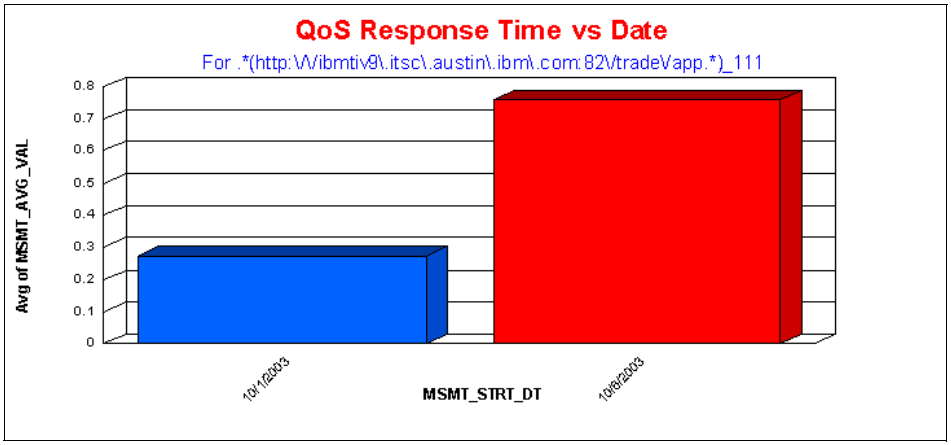


Figure 10-37 Response time for trade by trade_qos_lis listening policy

Part 4

Appendixes

© Copyright IBM Corp. 2003. All rights reserved.

A

Patterns for e-business

IBM Patterns for e-business is a set of proven architectures that have been compiled from more than 20,000 successful Internet-based engagements. This repository of assets can be used by companies to facilitate the development of Web-based applications. They help an organization understand and analyze complex business problems and break them down into smaller, more manageable functions that can then be implemented using low-level design patterns.

© Copyright IBM Corp. 2003. All rights reserved.

Introduction to Patterns for e-business

As companies compete in the e-business marketplace, they find that they must re-evaluate their business processes and applications so that their technology is not limited by time, space, organizational boundaries, or territorial borders. They must consider the time it takes to implement the solution, as well as the resources (people, money, and time) they have at their disposal to successfully execute the solution. These challenges, coupled with the integration issues of existing legacy systems and the pressure to deliver consistent high-quality service, present a significant undertaking when developing an e-business solution.

In an effort to alleviate the tasks involved in defining an e-business solution, IBM has built a repository of patterns to simplify the effort. In simple terms, a *pattern* can be defined as a model or plan used as a guide in making things. As such, patterns serve to facilitate the development and production of things. Patterns codify the repeatable experience and knowledge of people who have performed similar tasks before. Patterns not only document solutions to common problems, but also point out pitfalls that should be avoided. IBM Patterns for e-business consists of documented architectural best practices. They define a comprehensive framework of guidelines and techniques that were actually used in creating architectures for customer engagements. The Patterns for e-business bridge the business and IT gap by defining architectural patterns at various levels, from Business patterns to Application patterns to Runtime patterns, enabling easy navigation from one level to the next. Each of the patterns (Business, Integration, Application, and Runtime) help companies understand the true scope of their development project and provide the necessary tools to facilitate the application development process, thereby allowing companies to shorten time to market, reduce risk, and most important, realize a more significant return on investment.

The core types of Patterns for e-business are:

- Business Patterns
- Integration Patterns
- Composite Patterns
- Application Patterns
- Runtime Patterns and matching product mappings

When a company takes advantage of these documented assets, they are able to reduce the time and risk involved in completing a project.

For example, a line-of-business (LOB) executive who understands the business aspects and requirements of a solution can use Business patterns to develop a high-level structure for a solution. Business patterns represent common business problems. LOB executives can match their requirements (IT and business

drivers) to Business patterns that have already been documented. The patterns provide tangible solutions to the most frequently encountered business challenges by identifying common interactions among users, business, and data.

Senior technical executives can use Application patterns to make critical decisions related to the structure and architecture of the proposed solution. Application patterns help refine Business patterns so that they can be implemented as computer-based solutions. Technical executives can use these patterns to identify and describe the high-level logical components that are needed to implement the key functions identified in a Business pattern. Each Application pattern would describe the structure (tiers of the application), placement of the data, and the integration (loosely or tightly coupled) of the systems involved.

Finally, solution architects and systems designers can develop a technical architecture by using Runtime patterns to realize the Application patterns. Runtime patterns describe the logical architecture that is required to implement an Application pattern. Solution architects can match Runtime patterns to existing environment and business needs. The Runtime pattern they implement establishes the components needed to support the chosen Application pattern. It defines the logical middleware nodes, their roles, and the interfaces among these nodes in order to meet business requirements. The Runtime pattern documents what must be in place to complete the application, but does not specify product brands. Determination of actual products is made in the product mapping phase of the patterns.

In summary, Patterns for e-business captures e-business approaches that have been tested and proven. By making these approaches available and classifying them into useful categories, LOB executives, planners, architects, and developers can further refine them into useful, tangible guidelines. The patterns and their associated guidelines enable the individual to start with a problem and a vision, find a conceptual pattern that fits this vision, define the necessary functional pieces that the application will need to succeed, and then actually build the application. Furthermore, the Patterns for e-business provides common terminology from a project's onset and ensures that the application supports business objectives, significantly reducing cost and risk.

The Patterns for e-business layered asset model

The Patterns for e-business approach enables architects to implement successful e-business solutions through the re-use of components and solution elements from proven, successful experiences. The Patterns approach is based on a set of layered assets that can be exploited by any existing development

methodology. These layered assets are structured so that each level of detail builds on the last. These assets include:

- Business patterns that identify the interaction between users, businesses, and data.
- Integration patterns that tie multiple Business patterns together when a solution cannot be provided based on a single Business pattern.
- Composite patterns that represent commonly occurring combinations of Business patterns and Integration patterns.
- Application patterns that provide a conceptual layout describing how the application components and data within a Business pattern or Integration pattern interact.
- Runtime patterns that define the logical middleware structure supporting an Application pattern. Runtime patterns depict the major middleware nodes, their roles, and the interfaces between these nodes.
- Product mappings that identify proven and tested software implementations for each Runtime pattern.
- Best-practice guidelines for design, development, deployment, and management of e-business applications.

These assets and their relationship to each other are shown in Figure A-1.

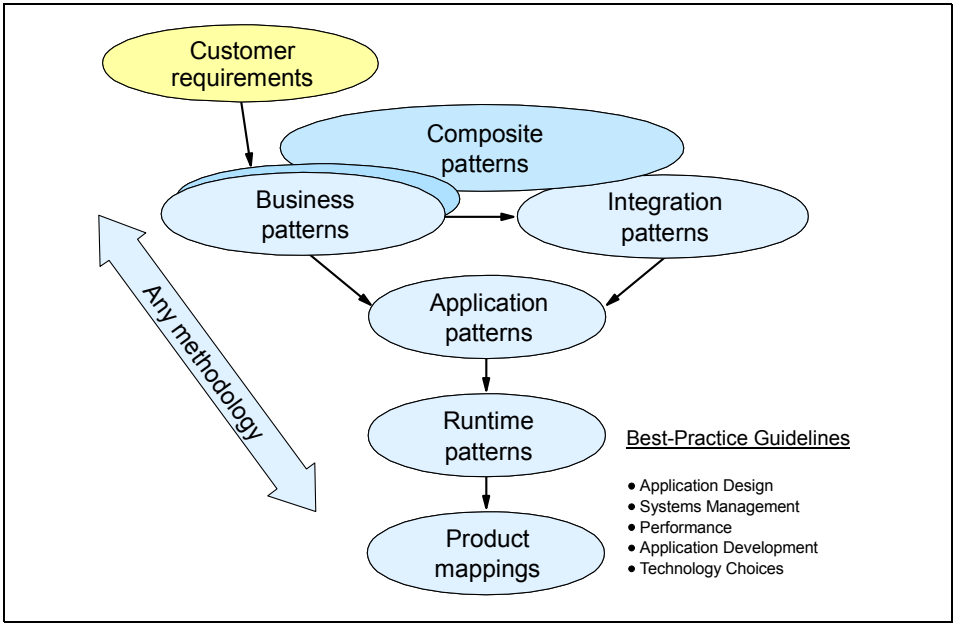


Figure A-1 Patterns layered asset model

Patterns for e-business Web site

The Patterns Web site provides an easy way of navigating top-down through the layered Patterns' assets in order to determine the preferred reusable assets for an engagement. For easy reference to Patterns for e-business, refer to the Patterns for e-business Web site at:

http://www.ibm.com/developerWorks/patterns/

How to use the Patterns for e-business

As described in the previous section, the Patterns for e-business are structured so that each level of detail builds on the last. At the highest level are Business patterns that describe the entities involved in the e-business solution. A Business pattern describes the relationship among the users, the business organization or applications, and the data to be accessed.

Composite patterns appear in the hierarchy above the Business patterns. However, Composite patterns are made up of a number of individual Business patterns and at least one Integration pattern. In this section, we discuss how to use the layered structure of the Patterns for e-business assets.

There are four primary Business patterns, as shown in Table A-1.

Table A-1 Business patterns

Business patterns	Description	Examples
Self-Service (user-to-business)	Applications where users interact with a business via the Internet	Simple Web site applications
Information Aggregation (user-to-data)	Applications where users can extract useful information from large volumes of data, text, images, and so on	Business intelligence, knowledge management, and Web crawlers
Collaboration (user-to-user)	Applications where the Internet supports collaborative work between users	E-mail, community, chat, video conferencing, and so on
Extended Enterprise (business-to-business)	Applications that link two or more business processes across separate enterprises	EDI, supply chain management, and so on

It would be very convenient if all problems fit nicely into these four Business patterns, but in reality things can be more complicated. The patterns assume that

all problems, when broken down into their most basic components, will fit more than one of these patterns. When a problem describes multiple objectives that fit into multiple Business patterns, the Patterns for e-business provide the solution in the form of Integration patterns.

Integration patterns enable us to tie together multiple Business patterns to solve a problem. The Integration patterns are shown in Table A-2.

Table A-2 Integration patterns

Integration patterns	Description	Examples
Access Integration	Integration of a number of services through a common entry point	Portals
Application Integration	Integration of multiple applications and data sources without the user directly invoking them	Message brokers and workflow managers

These Business and Integration patterns can be combined to implement installation-specific business solutions. We call this a Custom design.

We can represent the use of a Custom design to address a business problem through an iconic representation, as shown in Figure A-2.

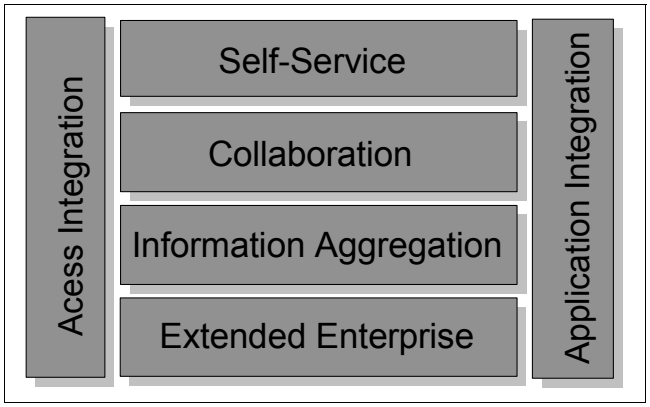


Figure A-2 Pattern representation of a Custom design

If any of the Business or Integration patterns are not used in a Custom design, we can show that with lighter blocks. For example, Figure A-3 on page 435 shows a Custom design that does not have a mandatory Collaboration business pattern or an Extended Enterprise business pattern for a business problem.

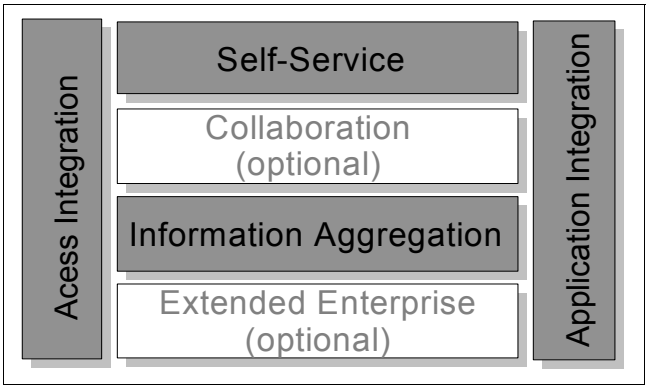


Figure A-3 Custom design

A Custom design may also be a Composite pattern if it recurs many times across domains with similar business problems. For example, the iconic view of a Custom design in Figure A-3 can also describe a Sell-Side Hub composite pattern.

Several common uses of Business and Integration patterns have been identified and formalized into Composite patterns, which are shown in Table A-3.

Table A-3 Composite patterns

Composite patterns	Description	Examples
Electronic Commerce	User-to-Online-Buying.	► www.macys.com ► www.amazon.com
Portal	Typically designed to aggregate multiple information sources and applications to provide uniform, seamless, and personalized access for its users.	► Enterprise intranet portal providing self-service functions, such as payroll, benefits, and travel expenses ► Collaboration providers who provide services such as e-mail or instant messaging
Account Access	Provide customers with around-the-clock account access to their account information.	► Online brokerage trading apps ► Telephone company account manager functions ► Bank, credit card, and insurance company online apps

Composite patterns	Description	Examples
Trading Exchange	Allows buyers and sellers to trade goods and services on a public site.	► Buyer's side: Interaction between buyer's procurement system and commerce functions of e-Marketplace ► Seller's side: Interaction between the procurement functions of the e-Marketplace and its suppliers
Sell-Side Hub (Supplier)	The seller owns the e-Marketplace and uses it as a vehicle to sell goods and services on the Web.	www.carmax.com (car purchase)
Buy-Side Hub (Purchaser)	The buyer of the goods owns the e-Marketplace and uses it as a vehicle to leverage the buying or procurement budget in soliciting the best deals for goods and services from prospective sellers across the Web.	www.wre.org (WorldWide Retail Exchange)

The makeup of these patterns is variable in that there will be basic patterns present for each type, but the Composite can easily be extended to meet additional criteria. For more information about Composite patterns, refer to *Patterns for e-business: A Strategy for Reuse* by Adams, et al.

Selecting Patterns and product mapping

After the appropriate Business pattern is identified, the next step is to define the high-level logical components that make up the solution and how these components interact. This is known as the Application pattern. A Business pattern will usually have multiple Application patterns identified that describe the possible logical components and their interactions. For example, an Application pattern may have logical components that describe a presentation tier for interacting with users, a Web application tier, and a back-end application tier.

The Application pattern requires an underpinning of middleware that is expressed as one or more Runtime patterns. Runtime patterns define functional nodes that represent middleware functions that must be performed.

After a Runtime pattern has been identified, the next logical step is to determine the actual product and platform to use for each node. Patterns for e-business have product mappings that correlate to the Runtime patterns, describing actual products that have been used to build an e-business solution for this situation.

Finally, guidelines assist you in creating the application using best practices that have been identified through experience.

For more information on determining how to select each of the layered assets, refer to the Patterns for e-business Web site at:

http://www.ibm.com/developerWorks/patterns/

B

Using Rational Robot in the Tivoli Management Agent environment

This appendix describes how to use Rational's Robot with a component of Tivoli Monitoring for Transaction Performance (TMTP), in order to measure typical end-user response times.

© Copyright IBM Corp. 2003. All rights reserved.

Rational Robot

Rational Robot is a functional testing tool that can capture and replay user interactions with the Windows GUI. In this respect, it is equivalent to Mercury's WinRunner, and we are using it to replace the function that was lost when we were forced to remove WinRunner from TAPM.

Robot can also be used to record and play back user interaction with a Java application, and with a Java applet that runs in a Web browser.

Documentation is included as PDF files in the note that accompanies this package.

Tivoli Monitoring for Transaction Performance (TMTP)

TMTP includes a component called Enterprise Transaction Performance (ETP). The core of ETP is an Application Response Measurement (ARM) agent, which recognizes "start" and "stop" calls made by an application (or script), and uses them to report response time and other data in real-time and historical graphs.

Since ETP is fully integrated with the Tivoli product set, thresholds can be set on response time, and TEC events can be created when the response time is too long. ETP saves its data in a database, from which it can be displayed with TDS, or sent to the Tivoli Data Warehouse and harvested using TSLA.

This way, standard capabilities of the TMTP/ETP product are used to measure response time, which can also be viewed in real-time graphs such as the one shown in Figure B-1 on page 441.

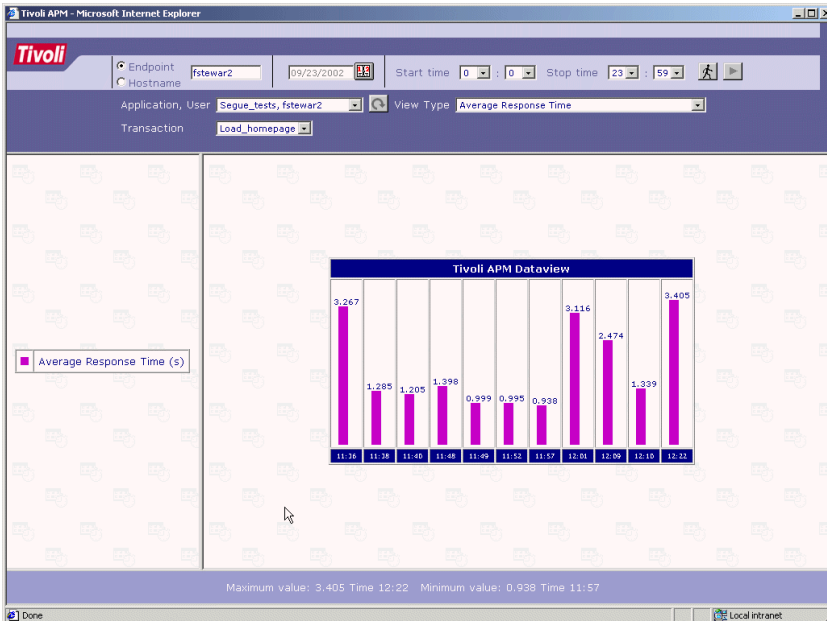


Figure B-1 ETP Average Response Time

In order for TMTP/ETP to record this data, the ARM API calls must be made from Rational Robot scripts.

The ARM API

The ARM API is an Open Group standard for a set of API calls that allow you to measure the performance of any application. The most common use of the API is to measure response time, but it can also be used to record application availability and account for application usage. The ARM API is documented at http://www.opengroup.org/management/arm.htm. The ARM Version 2 implementation is a set of C API calls, as shown in Figure B-2 on page 442.

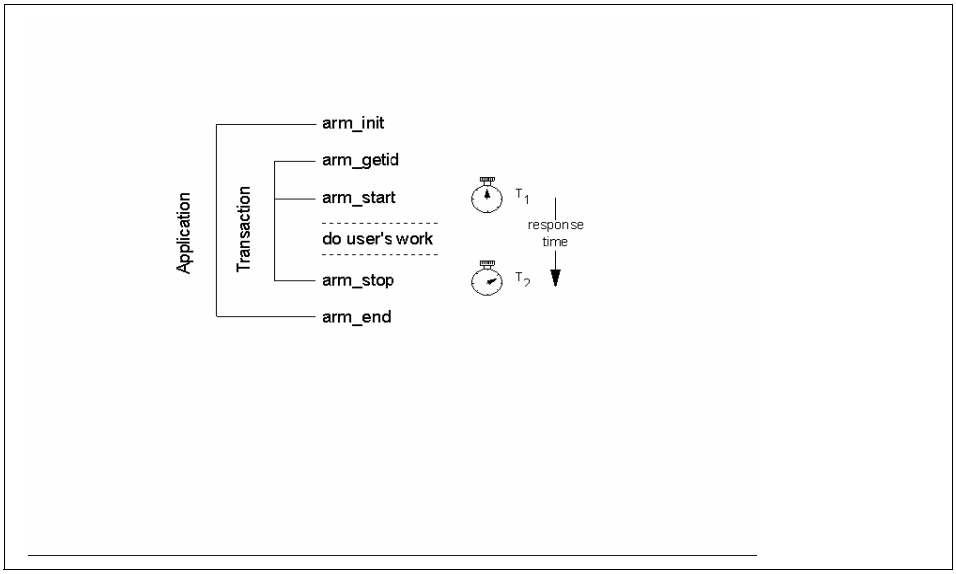


Figure B-2 ARM API Calls

There are six ARM API calls:

arm_init	This is used to define an application to the response time agent.
arm_getid	This is used to define a transaction to the response time agent. A transaction is always a child of an application.
arm_start	This call is used to start the response time clock for the transaction.
arm_update	This call is optional. It can be used to send a heartbeat to the response time agent, while the transaction is running. You might want to code this call in a long-running transaction, to receive confirmations that it is still running.
arm_stop	This call is used to stop the response time clock when a transaction completes.
arm_end	This call ends collection on the application. It is effectively the opposite of the arm_getid and arm_init calls.

The benefit of using ARM is that you can place the calls that start and stop the response time clock in exactly the parts of the script that you want to measure. This is done by defining individual applications and transactions within the script, and placing the ARM API calls at transaction start and transaction end.

Initial install

This is fairly straightforward: just run the setup executable and follow the "typical" install path. You will need to import the license key, either at the beginning of the install, or once the install has completed, using the Rational License Key Administrator, which should appear automatically.

At the end of the install, you will be prompted to set up the working environment for projects.

Decide on the location of your project. Before proceeding, open Windows Explorer and create the top-level directory of the project. Make sure the directory is empty. An example is shown in Figure B-3.

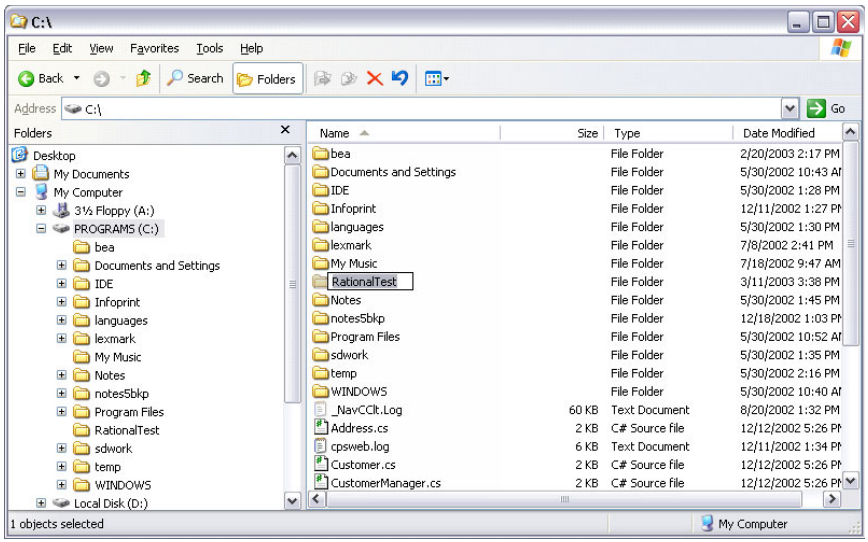


Figure B-3 Rational Robot Project Directory

To create a Rational project, perform the following steps:

1. Start the Rational Administrator by selecting **Start** → **Programs** → **Rational Robot** → **Rational Administrator**.
2. Start the New Project Wizard by selecting **File** → **New Project** on the Administrator menu. A window similar to Figure B-4 on page 444 should appear.

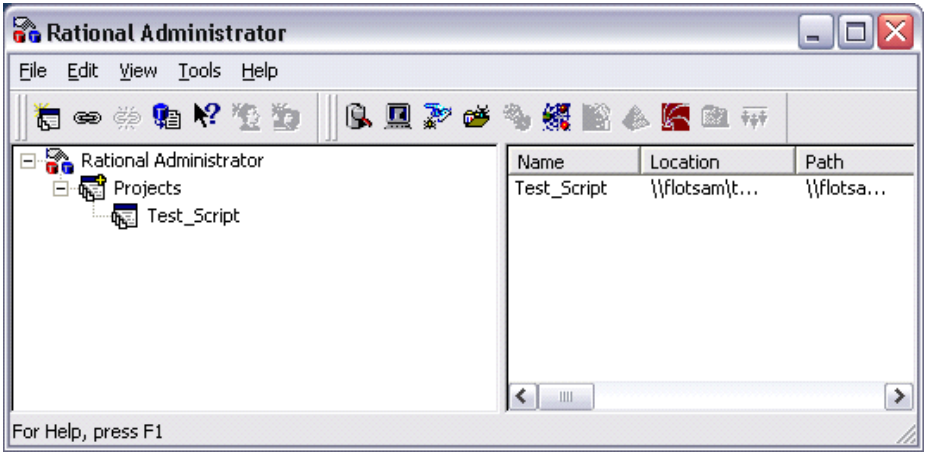


Figure B-4 Rational Robot Project

3. On the wizard's first page (Figure B-5 on page 445):
   a. Supply a name for your project, for example, testscripts. The dialog box prevents you from typing illegal characters.
   b. In the Project Location field, specify a UNC path to the root of the project, referring to the directory name you created in above. It does not really have to be a shared network directory with a UNC path.

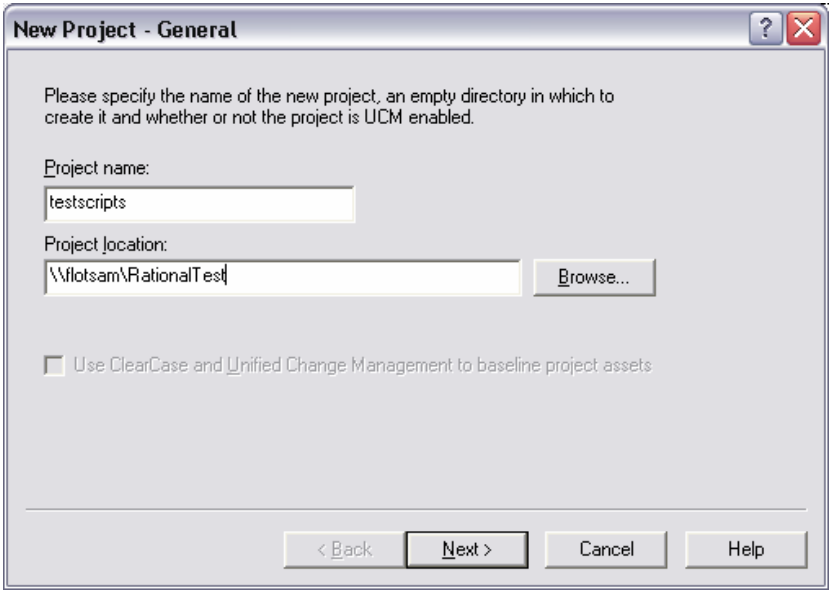

New Project - General
Please specify the name of the new project, an empty directory in which to create it and whether or not the project is UCM enabled.
Project name:
testscripts
Project location:
\\flotsam\RationalTest
Browse...
Use ClearCase and Unified Change Management to baseline project assets
< Back
Next >
Cancel
Help

Figure B-5 Rational Robot Project

4. Click **Next**. If you do create a password for the Rational project, supply the password on the Security page (Figure B-6 on page 446). If you do not create a password, then leave the fields blank on this page.

New Project - Security
Each Rational Project may be associated with a password. This password is required to connect to, configure or delete the project.
Password:
Confirm:
< Back
Next >
Cancel
Help

Figure B-6 Configuring project password

5. Click **Next** on the Summary page, and select **Configure Project Now** (Figure B-7 on page 447). The Configure Project dialog box appears (Figure B-8 on page 448).

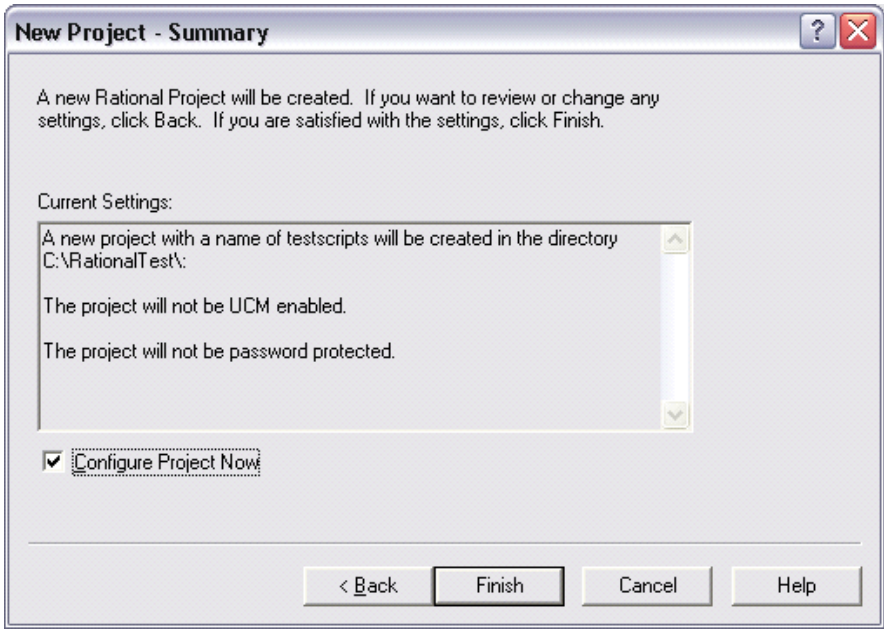

New Project - Summary
A new Rational Project will be created. If you want to review or change any settings, click Back. If you are satisfied with the settings, click Finish.
Current Settings:
A new project with a name of testscripts will be created in the directory C:\RationalTest\:
The project will not be UCM enabled.
The project will not be password protected.
Configure Project Now
< Back
Finish
Cancel
Help

Figure B-7 Finalize project

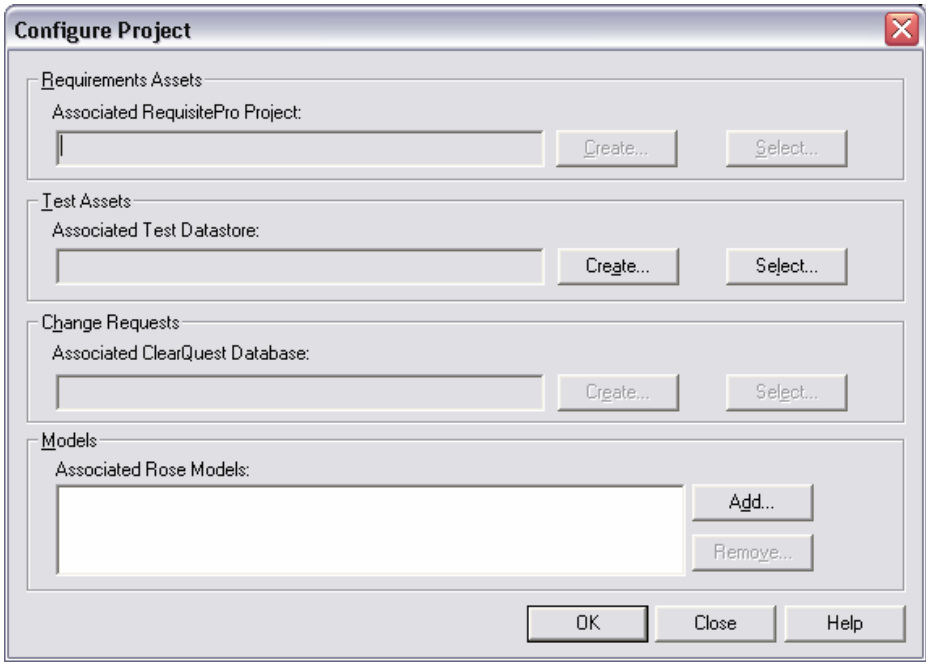


Figure B-8 Configuring Rational Project

A Rational Test datastore is a collection of related test assets, including test scripts, suites, datapools, logs, reports, test plans, and build information.

You can create a new Test datastore or associate an existing Test datastore.

For testing of Rational Robot, the user must set up the Test datastore.

To create a new test datastore:

1. On the Configure Project dialog box, click **Create** in the Test Assets area. The Create Test Datastore tool appears (Figure B-9 on page 449).

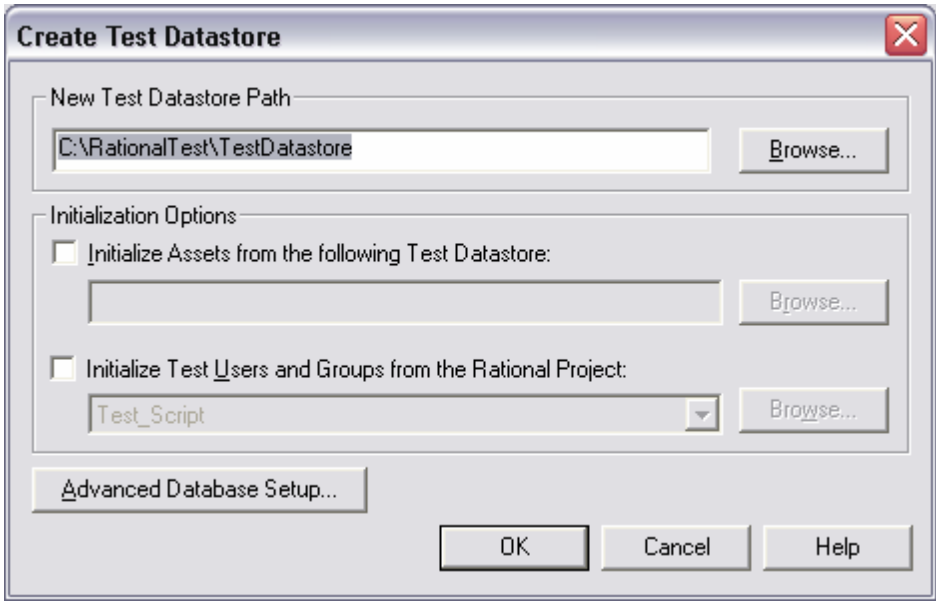


Figure B-9 Specifying project datastore

2. In the Create Test Datastore dialog box:
   a. In the New Test Datastore Path field, use a UNC path name to specify an area where you would like the tests to reside.
   b. Select initialization options as appropriate.
   c. Click Advanced Database Setup and select the type of database engine for the Test datastore.
   d. Click **OK**.

Working with Java Applets

If you are going to use Robot with Java applets, follow these simple instructions.

By default, Java testing is disabled in Robot. To enable Java testing, you need to run the Java Enabler. The Java Enabler is a wizard that scans your hard drive looking for Java environments, such as Web browsers and Sun JDK installations, that Robot supports. The Java Enabler only enables those environments that are currently installed.

If you install a new Java environment, such as a new release of a browser or JDK, you must rerun the Enabler after you complete the installation of the Java

environment. You can download updated versions of the Java Enabler from the Rational Web site whenever support is added for new environments. To obtain the most up-to-date Java support, simply rerun the Java Enabler.

Running the Java Enabler

1. Make sure that Robot is closed.
2. Select **Start** → **Programs** → **Rational Robot** → **Rational Test** → **Java Enabler**.
3. Select one of the available Java enabling types.
4. Select the environments to enable.
5. Click **Next**.
6. Click **Yes** to view the log file.

Note: If the Java Enabler does not find your environment, you must upgrade to one of the supported versions and rerun the Java Enabler. For a list of supported environments, see Supported Foundation Class Libraries link under the program's Help menu.

Using the ARM API in Robot scripts

Rational Robot uses the SQABasic script language, which is a superset of Visual Basic. Since the ARM API is a set of C functions, these functions must be declared to Robot before they can be used to define measurement points in SQABasic scripts.

This is best illustrated with an example. We used Robot to record a simple user transaction: opening Windows Notepad, adding some text, and closing the window. This created the following script, which contains the end user actions:

```
Sub Main
    Dim Result As Integer

    'Initially Recorded: 1/31/2003  4:12:02 PM
    'Script Name: test1

    Window SetContext, "Class=Shell_TrayWnd", ""
    Toolbar Click, "ObjectIndex=2;\;ItemText=Notepad", "Coords=10,17"

    Window SetContext, "Caption=Untitled - Notepad", ""
    InputKeys "hello"
    MenuSelect "File->Exit"
```

```
Window SetContext, "Caption=Notepad", ""
PushButton Click, "Text=No"

End Sub
```

We added the following code to the script.

1. Load the DLL in which the ARM API functions reside, and declare those functions. This must be done right at the top of the script. Note that the first line here is preceded by a single quote; it is a comment line.

```
'Declare ARM API functions. arm_update is not declared, since TMTP doesn't
use it.
    Declare Function arm_init Lib "libarm32" (ByVal appl_name As
String,ByVal appl_userid As String,ByVal flags As Long,ByVal data As
String,ByVal data_size As Long) As Long
    Declare Function arm_getid Lib "libarm32" (ByVal appl_id As Long,ByVal
tran_name As String,ByVal tran_detail As String,ByVal flags As Long,ByVal
data As String,ByVal data_size As Long) As Long
    Declare Function arm_start Lib "libarm32" (ByVal tran_id As Long,ByVal
flags As Long,ByVal data As String,ByVal data_size As Long) As Long
    Declare Function arm_stop Lib "libarm32" (ByVal start_handle As
Long,ByVal tran_status As Long,ByVal flags As Long,ByVal data As
String,ByVal data_size As Long) As Long
    Declare Function arm_end Lib "libarm32" (ByVal appl_id As Long,ByVal
flags As Long,ByVal data As String,ByVal data_size As Long) As Long
```

2. Then declare variables to hold the returns from the ARM API calls. Again, note the comment line preceded by a single quote mark.

```
'Declare variables to hold returns from ARM API calls
    Dim appl_handle As Long
    Dim getid_handle As Long
    Dim start_handle As Long
    Dim stop_rc As Long
    Dim end_rc As Long
```

3. Next, we added the ARM API calls to the script. Note that even though they are C functions, they are not terminated with a semicolon.

```
'Make ARM API setup calls, and display the return from each one.

    appl_handle = arm_init("Rational_tests","*",0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_init call is: " & appl_handle

    getid_handle = arm_getid(appl_handle,"Notepad","Windows",0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_getid call is: " & getid_handle

    'Start clock
```

```
    start_handle = arm_start(getid_handle,0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_start call is: " & start_handle
```

The arm_init and arm_getid calls define the application and transaction name. The application name used must match what is set up for collection in TMTP.

The arm_start call is used to start the response time clock, just before the transaction starts.

4. Finally, after the business transaction steps, we added the following:

```
'Stop clock
    stop_rc = arm_stop(start_handle,0,0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_stop call is: " & stop_rc

    'Make ARM API cleanup call
    end_rc = arm_end(appl_handle,0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_end call is: " & end_rc
```

The arm_stop call is made after the transaction completes.

The arm_end call is used to clean up the ARM environment, at the end of the script.

For the purposes of testing, we used MsgBox statements to display the return of each of the ARM API calls. The returns should be:

arm_init	positive integer
arm_getid	positive integer
arm_start	positive integer
arm_stop	0 (zero)
arm_end	0 (zero)

In production, you will want to comment out these MsgBox statements.

Here is the script file that we ended up with:

```
'Version 1.1 - Some declarations modified


'Declare ARM API functions. arm_update is not declared, since TMTP doesn't use
it.
    Declare Function arm_init Lib "libarm32" (ByVal appl_name As String,ByVal
appl_userid As String,ByVal flags As Long,ByVal data As String,ByVal data_size
As Long) As Long
    Declare Function arm_getid Lib "libarm32" (ByVal appl_id As Long,ByVal
tran_name As String,ByVal tran_detail As String,ByVal flags As Long,ByVal data
As String,ByVal data_size As Long) As Long
```

```
    Declare Function arm_start Lib "libarm32" (ByVal tran_id As Long,ByVal
flags As Long,ByVal data As String,ByVal data_size As Long) As Long
    Declare Function arm_stop Lib "libarm32" (ByVal start_handle As Long,ByVal
tran_status As Long,ByVal flags As Long,ByVal data As String,ByVal data_size As
Long) As Long
    Declare Function arm_end Lib "libarm32" (ByVal appl_id As Long,ByVal flags
As Long,ByVal data As String,ByVal data_size As Long) As Long


   Sub Main
    Dim Result As Integer

    'Initially Recorded: 1/31/2003  4:12:02 PM
    'Script Name: test1


    'Declare variables to hold returns from ARM API calls
    Dim appl_handle As Long
    Dim getid_handle As Long
    Dim start_handle As Long
    Dim stop_rc As Long
    Dim end_rc As Long


    'Make ARM API setup calls, and display the return from each one.

    appl_handle = arm_init("Rational_tests","*",0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_init call is: " & appl_handle

    getid_handle = arm_getid(appl_handle,"Notepad","Windows",0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_getid call is: " & getid_handle

    'Start clock
    start_handle = arm_start(getid_handle,0,"0",0)
    'Remove line below when you put this in production
    MsgBox "The return value from the arm_start call is: " & start_handle


    'Window SetContext, "Class=Shell_TrayWnd", ""
    'Toolbar Click, "ObjectIndex=2;\;ItemText=Notepad", "Coords=10,17"

    'Window SetContext, "Caption=Untitled - Notepad", ""
    'InputKeys "hello"
    'MenuSelect "File->Exit"

    'Window SetContext, "Caption=Notepad", ""
    'PushButton Click, "Text=No"
```

```
'Stop clock
stop_rc = arm_stop(start_handle,0,0,"0",0)
'Remove line below when you put this in production
MsgBox "The return value from the arm_stop call is: " & stop_rc

'Make ARM API cleanup call
end_rc = arm_end(appl_handle,0,"0",0)
'Remove line below when you put this in production
MsgBox "The return value from the arm_end call is: " & end_rc


End Sub
```

Scheduling execution of the script

You will probably want to run the script at regular intervals throughout the day. There is no standard way to schedule this using the ETP component of TMTP, but you can do it quite easily using the local scheduler in Windows NT/2000. The NT Task Scheduler was introduced with an NT 4.0 Service Pack or Internet Explorer 5.01, but on Windows 2000 systems, it is typically already installed.

The Windows scheduler can be set up using the command line interface, but it is easier and more flexible to use the graphical Task Scheduler utility, which you can find in the Windows Control Panel as the Scheduled Tasks icon (see Figure B-10).

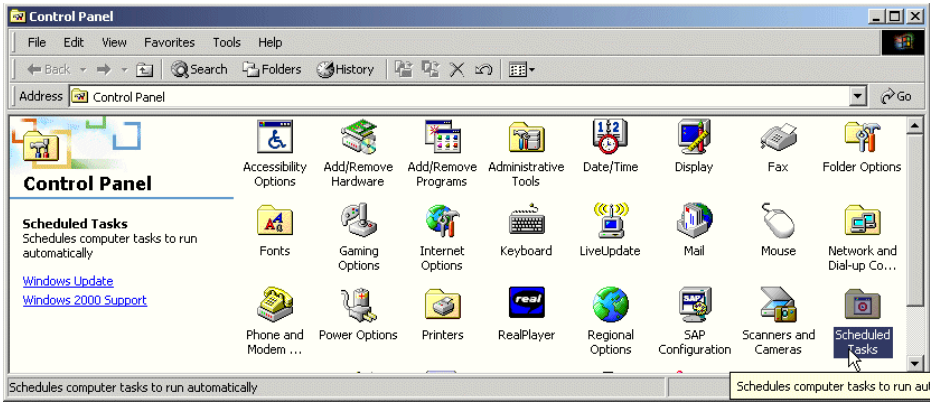


Figure B-10 Scheduler

A wizard will guide you through the addition of a new scheduled task. Select **Rational Robot** as the program you want to run (Figure B-11 on page 455).

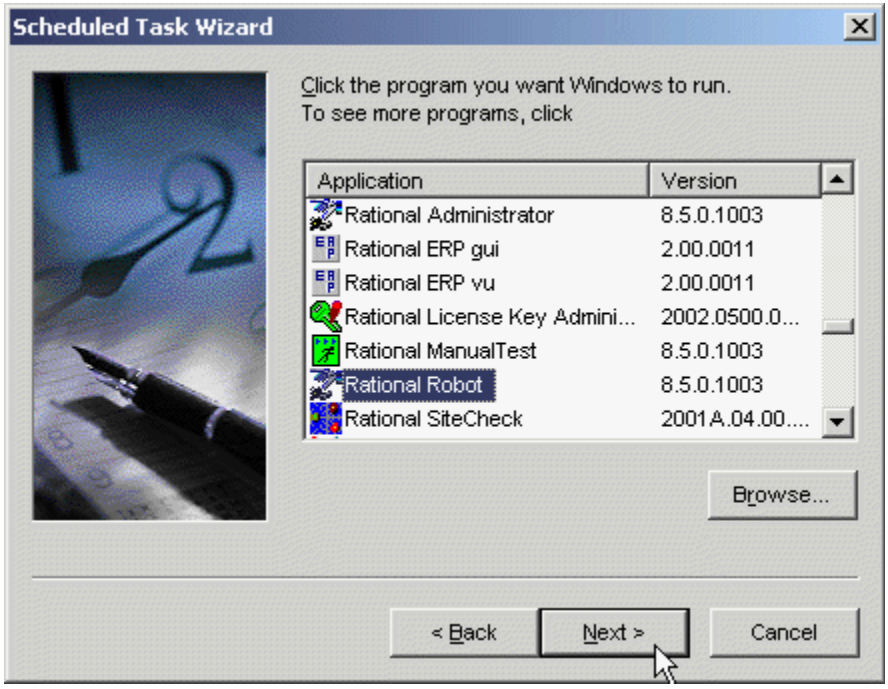


Figure B-11 Scheduling wizard

Name the task and set it to repeat daily (Figure B-12 on page 456). You can set how often it repeats during the day later.

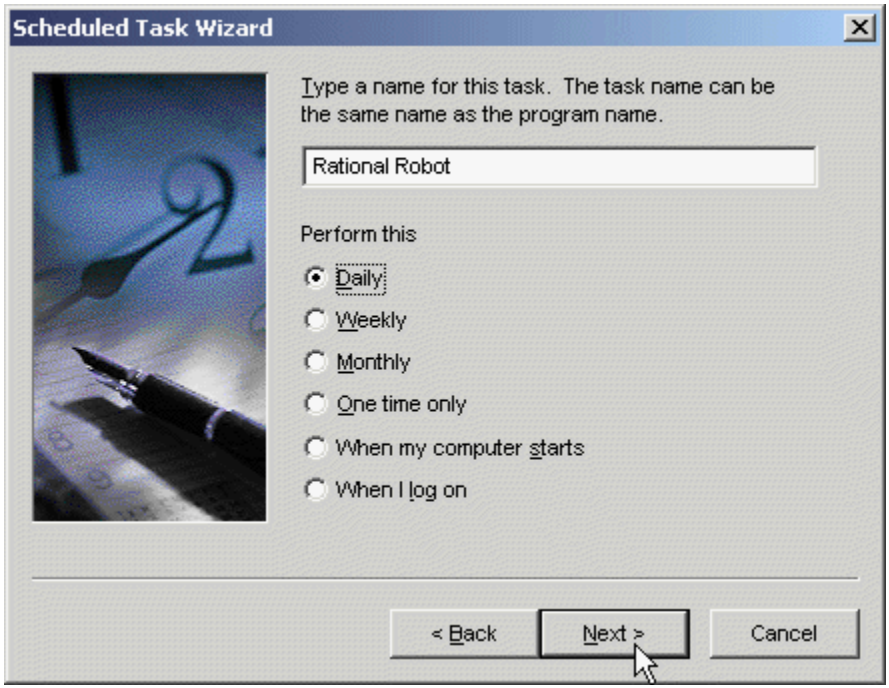


Figure B-12 Scheduler frequency

Set up the start time and date (Figure B-13 on page 457).

Figure B-13 Schedule start time

The task will need to run with the authority of some user ID on the machine, so enter the relevant user ID and password (Figure B-14 on page 458).

Scheduled Task Wizard
Enter the name and password of a user. The task will run as if it were started by that user.
Enter the user name:
FSTEWAR2\Administrator
Enter the password:
****
Confirm password:
****
< Back
Next >
Cancel

Figure B-14 Schedule user

Check the box in the window shown in Figure B-15 on page 459 in order to get to the advanced scheduling options.

Scheduled Task Wizard
You have successfully scheduled the following task:
Rational Robot
Windows will perform this task:
At 8:00 AM every day, starting 3/11/2003
Open advanced properties for this task when I click Finish.
Click Finish to add this task to your Windows schedule.
< Back
Finish
Cancel

Figure B-15 Select schedule advanced properties

Edit the contents of the Run option to use the Robot command line interface. For example:

```
"C:\Program Files\Rational\Rational Test\rtrobo.exe" ARM_example /user Admin
/project C:\TEMP\rationaltest\ScriptTest.rsp /play /build Build 1 /nolog /close
```

Details of the command line options can be found in the Robot Help topic, but are also included at the end of this document.

Set the **Start in** directory to the installation location; typically, this is Program Files\Rational\Rational Test (see Figure B-16 on page 460).

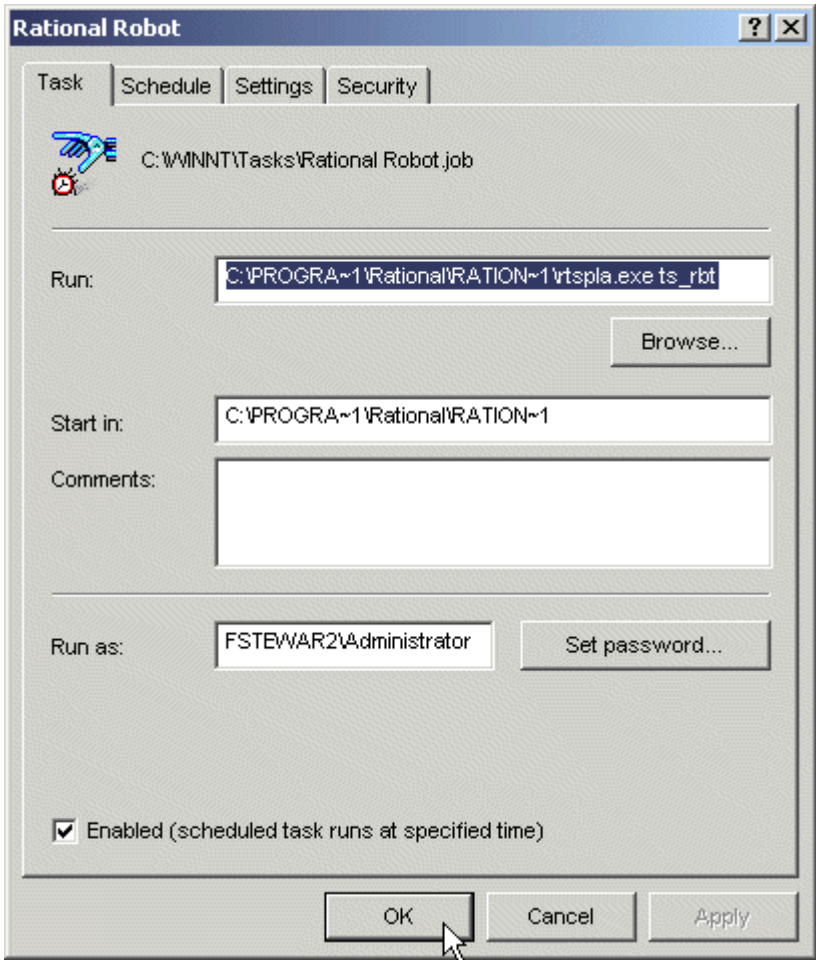


Figure B-16 Enable scheduled task

Select the **Schedule** tab and click on the **Advanced** button (see Figure B-17 on page 461).

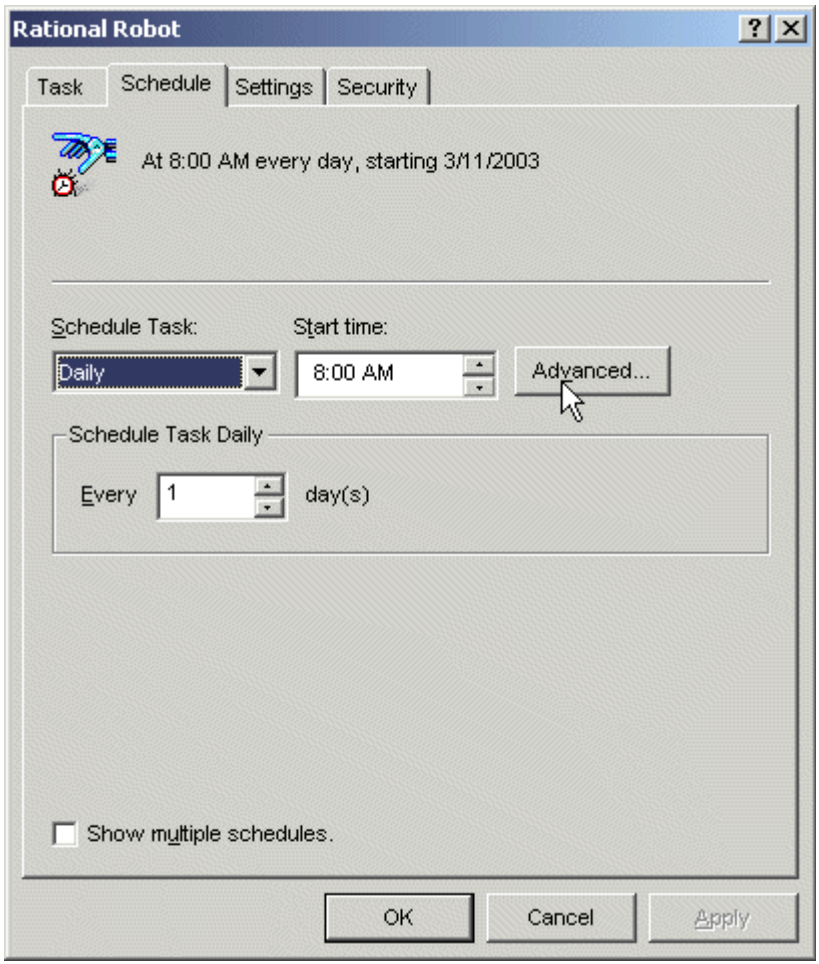


Figure B-17 Viewing schedule frequency

You can schedule the task to run every 15 minutes and set a date on which it will stop running (see Figure B-18 on page 462).

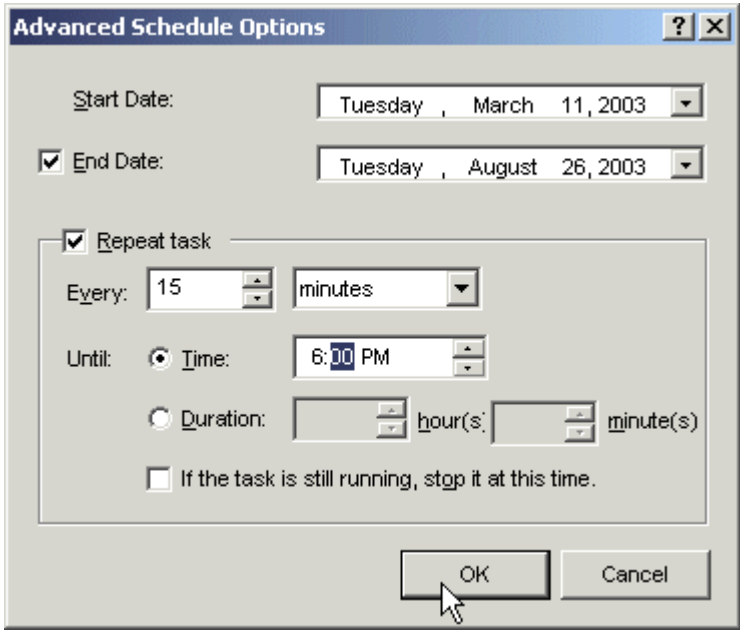


Figure B-18 Advanced scheduling options

It is also possible to schedule the execution of the Rational Robot using other framework functionality, such as scheduled Tivoli Tasks or custom monitors. These other mechanisms may have the benefit of allowing schedules to be managed centrally.

Rational Robot command line options

You can use the Rational Robot command line options to log in, open a script, and play back the script. The syntax is as follows:

```
rtrobo.exe [scriptname] [/user userid] [/password password] [/project full path
and full projectname] [/play] [/purify] [/quantify] [/coverage][/build build]
[/logfolder foldername] [/log logname] [/nolog] [/close]
```

The options are defined in Table B-1.

Table B-1 Rational Robot command line options

Syntax element	Description
rtrobo.exe	Rational Robot executable file.
scriptname	Name of the script to run.

Syntax element	Description
/user user ID	User name for login.
/password password	Optional password for login. Do not use this parameter if there is no password.
/project full path and full projectname	Name of the project that contains the script referenced in scriptname preceded by its full path.
/play	If this keyword is specified, plays the script referenced in scriptname. If not specified, the script opens in the editor.
/purify	Used with /play. Plays back the script referenced in scriptname under Rational Purify®.
/quantify	Used with /play. Plays back the script referenced in scriptname under Rational Quantify®.
/coverage	Used with /play. Plays back the script referenced in scriptname under Rational PureCoverage®.
/build build	Name of the build associated with the script.
/logfolder foldername	The name of the log folder where the test log is located. The log folder is associated with the build.
/log logname	The name of the log
/nolog	Does not log any output while playing back the script.
/close	

Some items to be aware of:

- Use a space between each keyword and between each variable.
- If a variable contains spaces, enclose the variable in quotation marks.
- Specifying log information on the command line overrides log data specified in the Log tab of the GUI Playback Options dialog box.

- If you intend to run Robot unattended in batch mode, be sure to specify the following options to get past the Rational Test Login dialog box:

  ```
  /user userid
  /password password
  /project full path and full projectname
  ```

  Also, when running Robot unattended in batch mode, you should specify the following options:

  ```
  /log logname  /build build  /logfolder foldername
  ```

An example of these options is as follows:

```
rtrobo.exe VBMenus /user admin  /project "C:\Sample Files\Projects\Default.rsp"
/play /build"Build1"/logfolder Default /log MyLog /close
```

In this example, the user admin opens the script VBMenus, which is in the project file Default.rsp located in the directory c:\Sample Files\Projects. The script is opened for playback, and then it is closed when playback ends. The results are recorded in the MyLog log located in the Default directory.

Obfuscating embedded passwords in Rational Scripts

Often, when recording Rational Scripts, it is necessary to record user IDs and passwords. This has the obvious security exposure that if your script is viewed, the password will be viewable in clear text. This section describes a mechanism for obfuscating the password in the script.

This mechanism relies on the use of an encryption library. The encryption library that we used is available on the redbook Web site. The exact link can be found in Appendix C, “Additional material” on page 473.

First, the encryption library must be registered with the operating system. For our encryption library, this was achieved by running the command:

```
regsvr32.exe EncryptionAlgorithms.dll
```

Once you have run this command, you must encrypt your password to a file for later use in your Rational Robot scripts. This can be achieved by creating a Rational Robot Script from the text in Example B-1 and then running the resulting script.

Example: B-1 Stashing obfuscated password to file

```
Sub Main

    Dim Result As Integer
    Dim bf As Object
    Dim answer As Integer
```

```
    ' Create the Encryption Engine and store a key
    Set bf = CreateObject("EncryptionAlgorithms.BlowFish")
    bf.key = "ibm"

    Begin Dialog UserDialog 180, 90, "Password Encryption"

        Text 10, 10, 100, 13, "Password: ", .lblPwd
        Text 10, 50, 100, 13, "Filename: ", .lblFile
        TextBox 10, 20, 100, 13, .txtPwd
        TextBox 10, 60, 100, 13, .txtFile

        OKButton  131, 8, 42, 13
        CancelButton  131, 27, 42, 13

    End Dialog

    Dim myDialog As UserDialog

DialogErr:
    answer = Dialog(myDialog)
    If answer <> -1 Then
        Exit Sub
    End If

    If Len(myDialog.txtPwd) < 3 then
        MsgBox "Password must have more than 3 characters!", 64, "Password
          Encryption"
        GoTo DialogErr
    End If

    ' Encrypt
    strEncrypt = bf.EncryptString(myDialog.txtPwd, "rational")


    ' Save to file
    'Open "C:\secure.txt" For Output Access Write As #1
    'Write #1, strEncrypt
    Open myDialog.txtFile For Output As #1
    If Err <> 0 Then
        MsgBox "Cannot create file", 64, "Password Encryption"
        GoTo DialogErr
    End If

    Print #1, strEncrypt
    Close #1

    If Err <> 0 Then
```

```
        MsgBox "An Error occurred while storing the encrypted password", 64,
          "Password Encryption"
        GoTo DialogErr
    End If

    MsgBox "Password successfully stored!", 64, "Password Encryption"

End Sub
```

Running this script will generate the pop-up window shown in Figure B-19, which asks for the password and name of a file to store the encrypted version of that password within.

Figure B-19 Entering the password for use in Rational Scripts

Once this script has run, the file you specified above will contain an encrypted version of your password. The password may be retrieved within your Rational Script, as shown in Example B-2.

Example: B-2 Retrieving the password

```
Sub Main
    Dim Result As Integer
    Dim bf As Object
    Dim strPasswd As String
    Dim fchar()
    Dim x As Integer

    ' Create the Encryption Engine and store a key
    Set bf = CreateObject("EncryptionAlgorithms.BlowFish")
    bf.key = "ibm"

    ' Open file and read encrypted password
    Open "C:\encryptedpassword.txt" For Input Access Read As #1
    Redim fchar(Lof(1))
```

```
    For x = 1 to Lof(1)-2
        fchar(x) = Input (1, #1)
        strPasswd = strPasswd & fchar(x)
    Next x

    ' Decrypt
    strPasswd = bf.DecryptString(strPasswd, "rational")
    SQAConsoleWrite "Decrypt: " & strPasswd


End Sub
```

The resulting unencrypted password has been retrieved from the encrypted file (in our case, we used the encryptedpassword.txt file) and placed into the variable strPasswd, and the variable may be used in place of the password where required. A complete example of how this may be used in a Rational Script is shown in Example B-3.

Example: B-3 Using the retrieved password

```
Sub Main

    'Initially Recorded: 10/1/2003  11:18:08 AM
    'Script Name: TestEncryptedPassword

    Dim Result As Integer
    Dim bf As Object
    Dim strPasswd As String
    Dim fchar()
    Dim x As Integer

    ' Create the Encryption Engine and store a key
    Set bf = CreateObject("EncryptionAlgorithms.BlowFish")
    bf.key = "ibm"

    ' Open file and read encrypted password
    Open "C:\encryptedpassword.txt" For Input Access Read As #1
    Redim fchar(Lof(1))
    For x = 1 to Lof(1)-2
        fchar(x) = Input (1, #1)
        strPasswd = strPasswd & fchar(x)
    Next x

    ' Decrypt the password into variable
    strPasswd = bf.DecryptString(strPasswd, "rational")

    Window SetContext, "Caption=Program Manager", ""
```

```
    ListView DblClick, "ObjectIndex=1;\;ItemText=Internet Explorer",
          "Coords=20,30"

    Window SetContext, "Caption=IBM Intranet - Microsoft Internet Explorer", ""
    ComboEditBox Click, "ObjectIndex=2", "Coords=61,5"
    InputKeys "http://9.3.4.230:9082/tmtpUI{ENTER}"
    InputKeys "root{TAB}^+{LEFT}"

    ' use the un-encrypted password retrieved from the encrypted file.
    InputKeys strPasswd

    PushButton Click, "HTMLText=Log On"
    Toolbar Click, "ObjectIndex=4;\;ItemID=32768", "Coords=20,5"
    PopupMenuSelect "Close"


End Sub
```

Rational Robot screen locking solution

Some users of TMTP have expressed a desire to be able to lock the screen while the Rational Robot is playing. The best and most secure solution to this problem is to lock the endpoint running simulations in a secure cabinet. There is no easy alternative solution, as the Rational Robot requires access to the screen context while it is playing back. During the writing of this redbook, we attempted a number of mechanisms to achieve this result, including use of Windows XP Switch User functionality, without success. The following Terminal Server solution implemented at one IBM customer site was suggested to us. We were unable to verify it ourselves, but we considered it useful information to provide as a potential solution to this problem.

This solution relies on the use of Windows Terminal Server, which is shipped with the Windows 2000 Server. When a user runs an application on Terminal Server, the application execution takes place on the server, and only the keyboard, mouse, and display information is transmitted over the network. This solution relies on running a Terminal Server Session back to the same machine and running the Rational Robot within the Terminal Server session. This allows the screen to be locked and the simulation to continue running.

1. Ensure that the Windows Terminal Server component is installed. If it is not, it can be obtained from the Windows 2000 Server installation CD from the Add On components dialog box (see Figure B-20 on page 469).

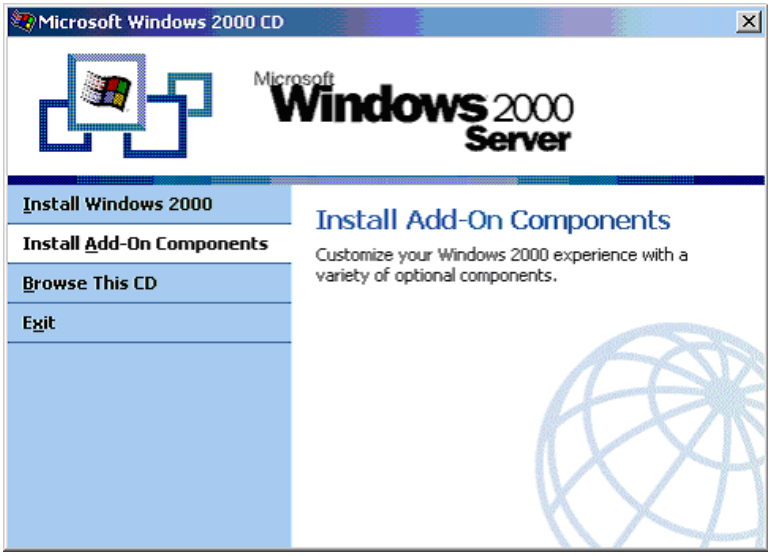


Figure B-20 Terminal Server Add-On Component

As the Terminal Server session will be back on the local machine, there is no reason to install the Terminal Server Licensing feature. Due to this fact, you should also select the Remote Administration mode option during Terminal Server install.

After the Terminal Server component is installed, you will need to reboot your machine.

2. Install the Terminal Server client on the local machine. The Terminal Server install provides a facility to create client installation diskettes. This same source can be used to install the Terminal Server client locally (Figure B-21 on page 470) by running the setup.exe (the path to this setup.exe is, by default, c:\winnt\system32\clients\tsclient\win32\disks\disk1).

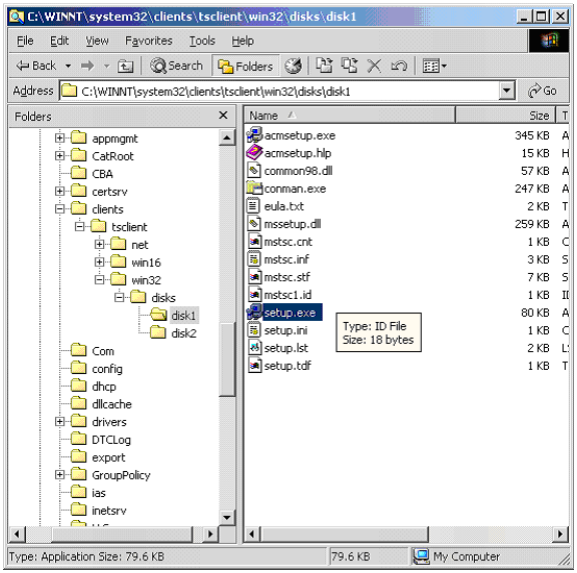


Figure B-21 Setup for Terminal Server client

3. Once you have installed the client, you may start a client session from the appropriate menu option. You will be presented with the dialog shown in Figure B-22 on page 471. From this dialog, you should select the local machine as the server you wish to connect to.

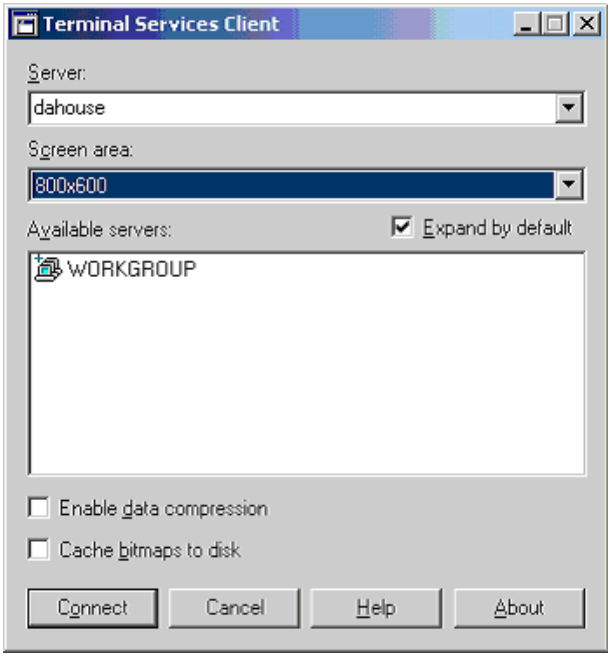


Figure B-22 Terminal Client Connection Dialog

> **Note:** It is useful to set the resolution to one lower than that used by the workstation you are connecting from. This allows the full Terminal Client session to be seen from the workstation screen.

4. Once you have connected, you will be presented with a standard Windows 2000 logon screen for the local machine within your client session. Log on as normal.
5. Now you can run your Rational Robot scripts using whichever method you would normally do this, with the exception of via GenWin. You may now lock the host screen and the Rational Robot will continue to run in the client session.

C

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246080
```

Alternatively, you can go to the IBM Redbooks Web site at:

```
ibm.com/redbooks
```

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246080.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
SG246080.zip	Zipped SQL statements and report samples

© Copyright IBM Corp. 2003. All rights reserved.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 1 MB
Operating System: Windows/UNIX
Processor: 700 or higher
Memory: 256 MB or more

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

The files in the zip archive are:

trade_petstore_method-avg.rpt
A sample Crystal Report file showing how to report on aggregated average data collected from TWH_CDW.

trade_petstore_method-max.rpt
A sample Crystal Report file showing how to report on aggregated maximum data collected from TWH_CDW.

cleancdw.sql The SQL script used to clean ITMTP source data from TWH_CDW.

resetsequences.sql The SQL script used to reset the ITMTP source ETL process.

Abbreviations and acronyms

ACF	Adapter Configuration Facility
AIX	Advanced Interactive Executive
AMI	Application Management Interface
AMS	Application Management Specifications
API	Application Programming Interface
APM	Application Performance Management
ARM	Application Response Measurement
ASP	Active Server Pages
BAROC	Basic Recorder of Objects in C
BDT	Bulk Data Transfer
BOC	Business Objects Container
CA	Certificate Authority
CGI	Common Gateway Interface
CICS	Customer Information Control System
CIM	Common Management Information
CLI	Command Line Interface
CMP	Container-Managed Persistence
CMS	Cryptographic Message Syntax
CPU	Central Processing Unit
CTS	Compatibility Testing Standard
DB2	Database 2™
DBCS	Double-byte Character Set
DES	Data Encryption Standard
DLL	Dynamic Link Library
DM	Tivoli Distributed Monitoring
DMTF	Distributed Management Task Force
DNS	Domain Name Service
DOM	Document Object Model
DSN	Data Source Name
DTD	Document Type Definition
EAA	Ephemeral Availability Agent (now QoS)
EJB	Enterprise Java Beans
EPP	End-to-End Probe platform
ERP	Enterprise Resource Planning
ETP	Enterprise Transaction Performance
GEM	Global Enterprise Manager
GMT	Greenwich Mean Time
GSK	Global Security Kit
GUI	Graphical User Interface
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	HTTP Secure
IBM	International Business Machines Corporation
IDEA	International Data Encryption Algorithm
IE	Microsoft Internet Explorer
IIOP	Internet Inter ORB Protocol
IIS	Internet Information Server
IMAP	Internet Message Access Protocol
IOM	Inter-Object Messaging
ISAPI	Internet Server API

© Copyright IBM Corp. 2003. All rights reserved.

ITM	IBM Tivoli Monitoring
ITMTP	IBM Tivoli Monitor for Transaction Performance
ITSO	International Technical Support Organization
JCP	Java Community Process
JDBC	Java Database Connectivity
JNI	Java Native Interface
JRE	Java Runtime Environment
JSP	Java Server Page
JVM	Java Virtual Machine
LAN	Local Area Network
LOB	Line of Business
LR	LoadRunner
MBean	Management Bean
MD5	Message Digest 5
MIME	Multi-purpose Internet Mail Extensions
MLM	Mid-Level Manager
ODBC	Open Database Connectivity
OID	Object Identifier
OLAP	Online Analytical Processing
OMG	Object Management Group
OOP	Object Oriented Programming
ORB	Object Request Broker
OS	Operating Systems
OSI	Open Systems Interconnection
PKCS10	Public Key Cryptography Standard #10
QoS	Quality of Service
RDBMS	Relational Database Management System
RIM	RDBMS Interface Module
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RTE	Remote Terminal Emulation
SAX	Simple API for XML
SDK	Software Developer's Kit
SHA	Secure Hash Algorithm
SI	Site Investigator
SID	System ID
SLA	Service Level Agreement
SLO	Service Level Objective
SMTP	Simple Mail Transfer Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
SSL	Secure Socket Layer
STI	Synthetic Transaction Investigator
TAPM	Tivoli Application Performance Management
TBSM	Tivoli Business Systems Manager
TCL	Terminal Control Language
TCP/IP	Transmission Control Protocol/Internet Protocol
TDS	Tivoli Decision Support
TEC	Tivoli Enterprise Console
TEDW	Tivoli Enterprise Data Warehouse
TIMS	Tivoli Internet Management Server
TMA	Tivoli Management Agent
TME	Tivoli Management Environment
TMR	Tivoli Management Region
TS	Transaction Simulation
TMTP	IBM Tivoli Monitor for Transaction Performance
UDB	Universal Database
UDP	User Datagram Protocol

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UUID	Universal Unique Identifier
VuGen	Virtual User Generator
VUS	Virtual User Script
Vuser	Virtual User
W3C	World Wide Web Consortium
WSC	Web Services Courier
WSI	Web Site Investigator
WTP	Web Transaction Performance
WWW	World Wide Web
XML	eXtensible Markup Language

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 482.

- *Deploying a Public Key Infrastructure*, SG24-5512
- *e-business On Demand Operating Environment*, REDP3673
- *IBM HTTP Server Powered by Apache on RS/6000*, SG24-5132
- *IBM Tivoli Monitoring Version 5.1: Advanced Resource Monitoring*, SG24-5519
- *Integrated Management Solutions Using NetView Version 5.1*, SG24-5285
- *Introducing IBM Tivoli Monitoring for Web Infrastructure*, SG24-6618
- *Introducing IBM Tivoli Service Level Advisor*, SG24-6611
- *Introducing Tivoli Application Performance Management*, SG24-5508
- *Introduction to Tivoli Enterprise Data Warehouse*, SG24-6607
- *Patterns for e-business: User to Business Patterns for Topology 1 and 2 Using WebSphere Advanced Edition*, SG24-5864
- *Planning a Tivoli Enterprise Data Warehouse Project*, SG24-6608
- *Servlet and JSP Programming with IBM WebSphere Studio and VisualAge for Java*, SG24-5755
- *Tivoli Application Performance Management Version 2.0 and Beyond*, SG24-6048
- *Tivoli Business Systems Manager: A Complete End-to-End Management Solution*, SG24-6202
- *Tivoli Business Systems Manager: An Implementation Case Study*, SG24-6032
- *Tivoli Enterprise Internals and Problem Determination*, SG24-2034
- *Tivoli NetView 6.01 and Friends*, SG24-6019
- *Tivoli Web Services Manager: Internet Management Made Easy*, SG24-6017

© Copyright IBM Corp. 2003. All rights reserved.

- *Tivoli Web Solutions: Managing Web Services and Beyond*, SG24-6049
- *Unveil Your e-business Transaction Performance with IBM TMTP 5.1,* SG24-6912
- *Using Databases with Tivoli Applications and RIM*, SG24-5112
- *Using Tivoli Decision Support Guides*, SG24-5506

Other resources

These publications are also relevant as further information sources:

- Adams, et al., *Patterns for e-business: A Strategy for Reuse*, MC Press, LLC, 2001, ISBN 1931182027
- *IBM Tivoli Monitoring for Transaction Performance: Enterprise Transaction Performance User's Guide Version 5.1*, GC23-4803
- *IBM Tivoli Monitoring for Transaction Performance Installation Guide Version 5.2.0*, SC32-1385
- *IBM Tivoli Monitoring for Transaction Performance User's Guide Version 5.2.0*, SC32-1386
- *IBM Tivoli Monitoring User's Guide Version 5.1.1*, SH19-4569
- *IBM Tivoli Monitoring for Web Infrastructure Apache HTTP Server User's Guide Version 5.1.0*, SH19-4572
- *IBM Tivoli Monitoring for Web Infrastructure Installation and Setup Guide Version 5.1.1*, GC23-4717
- *IBM Tivoli Monitoring for Web Infrastructure Reference Guide Version 5.1.1*, GC23-4720
- *IBM Tivoli Monitoring for Web Infrastructure WebSphere Application Server User's Guide Version 5.1.1*, SC23-4705
- *Tivoli Application Performance Management Release Notes Version 2.1*, GI10-9260
- *Tivoli Application Performance Management: User's Guide Version 2.1*, GC32-0415
- *Tivoli Decision Support Administrator Guide Version 2.1.1*, GC32-0437
- *Tivoli Decision Support Installation Guide Version 2.1.1*, GC32-0438
- *Tivoli Decision Support for TAPM Release Notes Version 1.1*, GI10-9259
- *Tivoli Decision Support User's Guide Version 2.1.1*, GC32-0436
- *Tivoli Enterprise Console Reference Manual Version 3.7.1*, GC32-0666
- *Tivoli Enterprise Console Rule Builder's Guide Version 3.7*, GC32-0669

- *Tivoli Enterprise Console User's Guide Version 3.7.1*, GC32-0667
- *Tivoli Enterprise Data Warehouse Installing and Configuring Guide Version 1.1*, GC32-0744
- *Tivoli Enterprise Installation Guide Version 3.7.1*, GC32-0395
- *Tivoli Management Framework User's Guide Version 3.7.1*, SC31-8434

The following publications come with their respective products and cannot be obtained separately:

- *NetView for NT Programmer's Guide Version 7*, SC31-8889
- *NetView for NT User's Guide Version 7*, SC31-8888
- *Web Console User's Guide*, SC31-8900

Referenced Web sites

These Web sites are also relevant as further information sources:

- Apache Web site

  http://www.apache.org/

- Computer Measurement Group Web site

  http://www.cmg.org/

- Crystal Decisions home page

  http://www.crystaldecisions.com/

- IBM DB2 Technical Support: All DB2 Version 7 FixPacks

  http://www-3.ibm.com/cgi-bin/db2www/data/db2/udb/winos2unix/support/v7fphist.d2w/report

- IBM Patterns for e-business

  http://www.ibm.com/developerWorks/patterns

- IBM Redbooks Web site

  http://www.redbooks.ibm.com

- IBM support FTP site

  ftp://ftp.software.ibm.com/software

- IBM Tivoli software support

  http://www.ibm.com/software/sysmgmt/products/support

- IBM WebSphere Application Server Trade3 Application

  http://www-3.ibm.com/software/webservers/appserv/benchmark3.html

- The Java Pet Store 1.3 Demo

  http://java.sun.com/features/2001/12/petstore13.html
- Java Web site for JNI documents

  http://java.sun.com/products/jdk/1.2/docs/guide/jni/
- The Object Management Group

  http://www.omg.org
- The Open Group

  http://www.opengroup.org
- OpenGroup ARM Web site

  http://www.opengroup.org/management/arm.htm
- Tivoli IBM Tivoli Monitoring for Transaction Performance Version 5.2 manuals

  http://publib.boulder.ibm.com/tividd/td/IBMTivoliMonitoringforTransactionPerformance5.2.html

How to get IBM Redbooks

You can order hardcopy Redbooks, as well as view, download, or search for Redbooks at the following Web site:

ibm.com/redbooks

You can also download additional materials (code samples or diskette/CD-ROM images) from that site.

IBM Redbooks collections

Redbooks are also available on CD-ROM. Click the CD-ROMs button on the Redbooks Web site for information about all CD-ROMs offered, as well as updates and formats.

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Numerics

3270 33, 80
  application 82
  transactions 35

A

administrator account 124, 133
agent 26
aggregate 34
  data 60, 66, 214
  topology 218
aggregation
  data 376
aggregation level 414
aggregation period 61
aggregation type 414
alerts 60
analysis 379
  historical 376
  multi-dimensional 417
  OLAP 379
  trend 417
application
  3270 82
  architecture 6
  design 32
  J2EE 5
  management 5–6
  patterns 436
  performance 7
  resource 26
  system 13
  tier 21
  transaction 5
  usefulness 32
applications
  source 378
architecture
  J2EE 7
ARM 33, 67, 257
  API 351, 441
  correlation 68
  engine 64–65, 67, 184
  records 188
authentication 76, 79
automated report 407
automatic
  baselining 213
  responses 168
automatic thresholding 240
availability 59, 154
  graph 222
  violation 219, 222
  Web transaction 35
Availability Management 18
avgWaitTime 163

B

back-end application tier 436
BAROC files 168
baselining
  automatic 213
BI
  See Busienss Intelligence
bidirectional interface 74
Big Board 212, 296
  filtering 215
  refresh rate 215
  view 44
bottleneck 205
breakdown 33, 220, 223
  STI transaction 220
  transaction 4, 35, 70
  transaction view 215
  view 215
Brio Technology 377
browser 59
brute force 195
business
  process 8
  system 30
Business Information Service 31
Business Intelligence 377
business intelligence reporting 379
Business Objects 377
BWM source information 192

© Copyright IBM Corp. 2003. All rights reserved.

BWM_c05_Upgrade_Processes 392
BWM_c05_Upgrade51_Process 400
BWM_c10_CDW_Process 400
BWM_DATA_SOURCE 399
BWM_m05_Mart_Process 400
BWM_TMTP_DATA_SOURCE 393
BWM_TWH_CDW_SOURCE 399
BWM_TWH_CDW_TARGET 400
BWM_TWH_MART_SOURCE 400
BWM_TWH_MART_TARGET 400
BWM_TWH_MD_TARGET 400

C

cache size 186
Capacity Management 18
categories
  reporting 409
cause
  problem 212
CDW
  See central data warehouse
central console 13
Central Data Warehouse 379
central data warehouse ETL 379
centralized
  management 365
  monitoring 14
certificate 77, 101, 179
Change Management 19
Client Capture 59
client-server 12
Cognos 377
collect performance data 157
Comments 224
common dimensions 376
Common Warehouse Metadata 377
component
  report 413
  service 16
confidentiality 76
configuration
  adapter 168
  DB2 91
  playback 371
  schedule 371
  SnF agent 77
  treshold 371
  WebSphere 91
Configuration Management 18
Configure Schedule 249
connection
  ODBC 393
connection pool 163
console
  central 13
consolidate 187
constraint 29
Contingency Planning 18
control heap size 385
controlled measurement 35
cookie 338
corrective action 13, 31, 168
correlate 168
correlating data 376
correlation 66, 225, 376
  engine 31
Cost Management 17
counters 157
create
  bufferpool 91
  database 91
  datastore 448
  depot directory 89
  discovery policy 261, 266
  file system 88
  listening policies 287
  listening policy 271
  new user 143
  Playback policy 251, 369
  realm 255
creating
  reports 407
Crystal Decisions 377
Crystal Reports 418
current data 60
custom registry 79
CWM
  See Common Warehouse Metadata

D

data
  aggregate 66
  aggregated 60, 214
  correlating 376
  event 214
  extract 378

gathering 34
historical 379, 392
histrorical 376
management 378
measurement 377
persistence 62
reference 33
data aggregation 376
data analysis 379
data gathering 382
data mart 191, 377, 379, 406
format 379
data mart database 381
data mart ETL 379, 381
data mining 379
data source 378, 393
ODBC 419
data target 378
data warehouse 379
database
central warehouse 380
data mart 381
warehouse source 380
datastore
create 448
DB2 112
fenced 144
instance 145
user 146
DB2 instance
64-bit 208
db2admin 143
db2start 178
db2stop 178
dbtmtp 113
debug 354
demilitarized zone 21, 24
demilitarized zone (DMZ) 82
deploying
GenWin 365
J2EE component 278
TMTP components 239, 310
details
policy 214
dimension tables 381
dimensions
common 376
discovery policy 228, 239
create 261, 266
discovery task 160
DMLinkJre 156
DNS 118
duplication 14
duration 214
dynamic data tables 382

E

ease-of-use 32
e-business
application 14, 38, 80
architecture 81
infrastructure 80
management 40
patterns 22
e-business performance 38
Edge Aggregation 71
effectiveness 204
EJB performance 163
encryption 356, 464
endpoint database 382
Endpoint Group 265
end-to-end view 376
Enterprise Application Integration 8
enterprise transaction 5, 33
Enterprise Transaction Performance 58
environment variable 106
ETL
central data warehouse 379
data mart 379, 381
process 394
processes 404
source 379
target 379
upgrade log files 392
ETL processes 380
ETL programs 378
ETL1
upgrade 392
ETL1 name 389
event 168–169, 224
class 168
data 214
notifications 31
view 216
event generation 240
exchange certificate 101
extract data 378

extreme case reports 413
extreme value 413

F

fact table 414
fact tables 381
filtering
  Big Board 215
format
  data mart 379
framework xxiii, 28, 60, 77
functionality 32

G

gathering
  data 34
gathering data 382
General
  report 296
general
  management 15
  topology 222
generating
  JKS files 93
  KDB files 98
  STH files 98
Generic Windows 229
  GenWin 233
GenWin 195, 363, 365, 471
  deploy 365
  limitations 234
  placing 80
  recording 233
ggregated
  correlation 71
graph
  QoS 213
  STI 213
GUI script 344
  record 345
guidelines 22

H

hacking 24
health
  monitoring policy 222
health check reports 408
heap size
  control 385
Help Desk 19
helper table 381
historical analysis 376
historical data 60, 170, 376, 379, 392
holes 164
host name 87, 132, 376
Host Socket Close 224
hosting 22
hostname 118
hotfix 332
hourly performance 220
HTTP
  request 230
  response code 230
hyperlink 218

I

IBM Automation Blueprint 30
icon status 212, 216
Idle Times 224
ikeyman 93
implementation 79
indications 166, 170–171
indicators 162
information
  page-specific 223
  transaction process 380
infrastructure
  management 10
  system management 26
installation
  Rational Robot 326
  Web Infrastructure 155
instance 66, 91
  data 60
  topology 47, 213
  transaction 217
instance owner 395
instrument 188
instrumentation 157
Integrated Solutions Console 174
integration 30
  point 51
interactive reporting 379
Internet zone 129
interpreted status 217

intranet 58
zone 130
IP address 376
IPCAppToEngSize 185

J

J2EE 229
application 5, 81
architecture 7
component 278
component remove 196
components 307
monitoring 72, 76, 82, 188, 232
support 73
topology 216
J2EE monitoring 293
settings 204
J2EE Monitoring Management Agent 82
J2EE subtransaction 293
Java Enabler 335
Java Management Extension 61
Java Management Extensions 9
Java Runtime Environment 156, 483
Java Virtual Machine 7
JDBC 206
error 178
JITI 74
probes 75
JKS 123
files 93
JSP errors 164
Just In Time Instrumentation 74
JVM 336, 365
memory 163

K

KDB files 98

L

layered assets 437
LDAP 25, 79
legacy systems 9, 11, 23, 81
License Key 333–334, 443
listening policy 189, 222, 239
create 271
load balance 22, 80–81
Local Socket Close 224
log files
ETL upfrade 392

M

MAHost 189
mail servers 59
managed application
create 158
objects 158
managed node 173
managed resource 166
management
application 5–6
general 15
needs 5, 13
specialized 15
Management Agent 247
deploying 311
redirect 181
management agent 57, 63, 365
communication with server 65
discovery 57
event support 65
installation 130
listening 58, 63
playback 58, 63
store and forward 58, 65
management data 378
Management Server 61, 63, 82, 247
custom installation 88, 107
placing 79
port number 140
typical installation 137
uninstall 193
MarProfile 60
Mask field 122
MBean 9, 63, 182–183
measurement 34
controlled 35
report 413
measurement data 377
metadata interchange 377
metrics
report 414
middleware 30
migration 193
Min/Max View 217
mission-critical 15

modules
warehouse 378
monitoring 15, 153
centralized 14
collection 60
proactive 154
profile 166
real-time 35, 171
monitoring policy 213, 239
health 222
multi-dimensional analysis 417
multidimensional reporting 406
multiple DMZ 77, 79
multiple firewall 38, 79

N

non-edge aggregation 71
Notes Servers 59

O

Object Management Group 377
object model store 62
occurrences 164, 170
ODBC
data source 419
ODBC connection 393
OLAP 375, 417
analysis 379
OLAP tools 406
On Demand Blueprint 28
Automation 28
Integration 28
Virtualization 28
oslevel 88
overall transactions
over time report 220
overview xxi, 55
topology 51
owner
instance 395

P

Page Analyzer
viewer 50
Page Analyzer Viewer 213, 223
Comments 224
events 224
Host Socket Close 224
Idle Times 224
Local Socket Close 224
Properties 224
Sizes 224
Summary 224
pages
visited 223
page-specific information 223
parent based correlation 69
path
transaction 212
pattern
e-business 22
Patterns for e-business 429
PAV report 213
performance 157, 338
EJB 163
hourly 220
measure 350
statistics 70
subtransaction 221
subtransactions 221
trace 70
violation 44–45, 208
performance data
collection 157
Pet Store application 307–308
playback 35, 326, 337, 347, 365, 440
monitoring tools 227
schedule 248
Playback Policy
create 369
Playback policy
create 251
playback Policy 222
playback policy 252
PMR 189–190
policies 32
policy
details 214
discovery 228
listening 222
management 64
monitoring 213
playback 222
region 158, 161
policy based correlation 69
Port

default 156
number 123, 132
predefined
action 168
rules 168
presentation
layer 24
tier 436
proactive monitoring 27, 154
probe 35, 59, 74
problem
cause 212
identification 35
resolution 154
Problem Management 19
process
ETL 394
processes
ETL 380, 404
product mappings 437
production
environment 87, 204
production status 404
Profile Manager 166
profile monitoring 166
Properties 224
protocol layer 326
provisioning 29
proxy 26, 121, 132
prune 191
public report 414

Q

QoS 229, 232
configuring 253
graph 213
placing 79
Quality of Service 229, 232, 257
deployment 259
Quality of Service Management Agent 82

R

Rational Robot 58–59, 195, 233, 440
installation 326
license key 333
Rational Robot/GenWin Management Agent 82
raw data 376
RDBMS 377
realm 255
create 255
settings 256
real-time 170
monitoring 8, 33, 35, 171
report 40, 62
realtime reporting 50
record 337, 440
GUI script 345
simulation 344
recording 35
Redbooks Web site 482
Contact us xxiv
reference
data 33
transaction 33
refresh rate 175
Big Board 215
register 368
remove
J2EE component 196
report
automatic 407
availability graph 222
categories 409
component 413
general topology 222
measurement 413
metrics 414
overall transactions over time 220
Page Analyzer Viewer 223
public 414
schedule 416
Slowest Transactions Table 222
summary 413
time inteval 416
transaction performance 295
Transaction with Subtransaction 221
types 295
Report Interface 379
TEDW 407
report interface
TEDW 381
reporting 34, 60
business intelligence 379
capabilities 44
interactive 379
multidimentional 406
roles 407

reports
creating 407
extreme case 413
health check 408
request
HTTP 230
requests
Web page 224
requirements
operating system 88
resolution
problem 154
resource
application 26
model 31, 60, 162, 168, 170
response
automatic 168
response code
HTTP 230
response time
transactions 163
Response Time View 218
response time view 217, 321
Retrieve Latest Data 213
reverse proxy 77, 80, 258
reverse-proxy 257
RI
See Report Interface
RMI 206
roles
reporting 407
root
account 123, 132
transaction 76, 217
root cause 225, 288
root cause analysis 306
Root-cause analysis 8
rules 168
ruleset 168–169
Runtime patterns 436

S

SAP 33, 80, 82
transaction 35
scalable 81
schedule 454
playback 248
report execution 416
screen lock 360, 468
secure zone 79
security 156, 170
features 76
protocol 76
TEDW 377
Seibel, 82
server
TEDW Control Center 388
virtual 261
server status 162
service 14–15
component 16
delivery 17
specialized 13
Service Level Management 17
sessions 163
setup wizard 329
severity
violation 216
severity codes 167
sibling transaction 70
simulation
transaction 35
single-point-of-failure 153
Sizes 224
slow transaction 217
Slowest Transactions Table 222
SMTP settings 176
SnF agent 77, 79
configuration 77
deployment 118
placing 79
redirect 181
SNMP
settings 175
trap 182
Software Control and Distribution 19
solution 14
source
data 393
warehouse 394
source applications 378
source ETL 379
specialized
management 15
services 13
SSL 110, 244
agent 140

setup 179
transaction 77
staging area tables 382
standardization 12
star schema 381, 408, 416
stash file 122
statistcis
performance 70
status
interpreted 217
production 404
server 162
STH files 98
STI 229–230, 241
graph 213
limitations 231
placing 80
Recorder 242
recording 248
subtransaction 219
STI transaction
breakdown 220
store and forward agent 77
Store and Forward Management Agent 82
subscribers 166
subtransaction 212
performance 221
selection 247
STI 219
times 212
Summary 224
summary report 413
surveillance 15, 60, 153
synchronization
time 71
Synthetic Transaction Investigator 229–230
Synthetic Transaction Investigator Management Agent 82
system event 62
system management 5, 28
infrastructure 26

T

table
dimension 381
fact 381, 414
helper 381
table space
temporary user 386
tables
dynamic data 382
staging area 382
target
warehouse 394
target ETL 379
task
discovery 160
TEC
adapter 167
events 440
TEDW
installation 387
installation user 386
Report Interface 407
security 377
user access 394
TEDW Central Data Warehouse 380
TEDW Control Center 380
TEDW Control Center server 388
TEDW report interface 381
TEDW repository 376
TEDW server 379
Terminal Server 360, 468
Test datastore 343
thread pool 163
threshold 167, 253
threshold setting 45, 64
threshold violation 68
thresholding 213
automatic 240
thresholds 61, 219
Thresholds View 217
tier
application 21
time interval
report 416
time synchronization 71
time zone 173
timer 350
Timer.goGet() 47
times
subtransaction 212
Tivoli Data Warehouse 191
Tivoli Enterprise Data Warehouse
source applications 378
Tivoli Internet Management Server (TIMS) 57
TMTP 389

application 149
database 149
ETL1 name 389
implementation 79
installation 85
port numbers 92
roles 79
TMTP components 40
Discovery component 40
J2EE monitoring component 43
Listening components 41
Playback components 41
Quality of Service component 42
Rational Robot/Generic Windows 43
Synthetic Transaction Investigator 43
TMTP_DB_Src 393
Tmw2kProfile 166
topology 212
aggregated 218
instance 47, 213
J2EE 216
overview 51
report 212, 216
view 44, 212, 215, 218–219, 296, 318
topology view 300
trace
performance 70
Trade3 application 236
transaction
3270 35
application 5
behaviour 212
breakdown 4, 35
control 24
decomposition 57
enterprise 5, 33
instance 217
path 212
reference 33
response time 163
root 76, 217
SAP 35
simulation 35
slow 217
type 4
Web 4, 33
worst performing 222
transaction process information 380
Transaction with Subtransaction 221, 297
report 317
transactions 408
Transactions With Subtransactions 49
transformation services 24
trend analysis 417
troubleshooting 188
TWH_CDW 380
TWH_MART 380
TWH_MD 380
TWHApp.log 391

U

upgrade 193
upgrade ETL1 392
upload 187
user
TEDW installation 386
temporaty table space 386
user access
TEDW 394
User interface. 61

V

value
extreme 413
variable 106
Verification Point 345, 347
adding 347
violation
availability 219, 222
percent 294
severity 216
virtual host 265
virtual server 261
visited
pages 223
VU script 345
VuGen 59

W

warehouse
cetral database 380
source 394
source database 380
target 394
warehouse modules 378
wcrtprf 166

wcrtprfmgr 166
wdmdistrib 156, 167
wdmeditprf 166
Web
  application tier 436
  Detailer 223
  transaction 33
Web Health Console 51, 60, 170, 217, 306
Web page
  activity 224
Web page requests 224
Web transaction 4, 33
  availability 35
Weblogic 201
  application server 307
WebSphere server
  start and stop 116
  stop and start 150
WriteNewEdge 190
wscp 156
wsub 166
wwebshpere 161

End-to-End e-business Transaction Management Made Easy

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

End-to-End e-business Transaction Management Made Easy

Redbooks

Seamless transaction decomposition and correlation

Automatic problem identification and baselining

Policy based transaction discovery

This IBM® Redbook will help you install, tailor, and configure the new IBM Tivoli Monitoring for Transaction Performance Version 5.2, which will assist you in determining the business performance of your e-business transactions in terms of responsiveness, performance, and availability.

The major enhancement in Version 5.2 is the addition of state-of-the-art industry strength monitoring functions for J2EE applications hosted by WebSphere® Application Server or BEA Weblogic. In addition, the architecture of Web Transaction Monitoring (WTP) has been redesigned to provide for even easier deployment, increased scalability, and better performance. Also, the reporting functions has been enhanced by the addition of ETL2s for the Tivoli Enterprise Date Warehouse.

This new version of IBM Tivoli® Monitoring for Transaction Performance provides all the capabilities of previous versions of IBM Tivoli Monitoring for Transaction Performance, including the Enterprise Transaction Performance (ETP) functions used to add transaction performance monitoring capabilities to the Tivoli Management Environment® (with the exception of reporting through Tivoli Decision Support). The reporting functions have been migrated to the Tivoli Enterprise Date Warehouse environment.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks

SG24-6080-00 ISBN 073849323